COLEÇÃO DE PROGRAMAS PARA

MSX

2ª EDIÇÃO

VOL. II

COLEÇÃO DE PROGRAMAS PARA MSX

VOL. II

2ª EDIÇÃO

COLEÇÃO DE PROGRAMAS PARA MSX

VOL. II

COORDENAÇÃO:

Renato da Silva Oliveira

CO-AUTORES:

Carlos Eduardo Rocha Salvato
Fernando da Costa Grossi
Henrique de Figueredo Luz
Luis Fernando Daniello Dias
Luiz Tarcísio de Carvalho Jr.
Milton Maldonado Jr.
Pierluigi Piazzi
Rubens Pereira Silva Jr.
Wilson Fazzio Martins
Yeh Yu Sung

EXPEDIENTE

Coordenação Editorial .... PIERLUIGI PIAZZI
Coordenação Didática .... BETTY FROMER PIAZZI
Editoração e Diagramação ..... RENATO S. OLIVEIRA
Arte .... ANA LÚCIA ANTICO
Capa .... ÁLVARO GUILLERMO
Ilustrações .... FERNANDO MORETTI
Produção .... ROSA K. FROMER

ALEPH
Publicações e Ass. Pedag. Ltda.
Av. Brig. Faria Lima 1451 c. 31
01451 S. Paulo SP
(011) **813-2033**

Dados de Catalogação na Publicação (CIP) Internacional
(Câmara Brasileira do Livro, SP, Brasil)

Coleção de programas para MSX / coordenação Renato
C655 da Silva Oliveira ; co-autores Carlos Eduardo
v.1-2 Rocha Salvato ... [et al.]. -- São Paulo : Aleph,
1986.
(Coleção MSX)

1. BASIC (Linguagem de programação para computadores) 2. MSX (Computadores) 3. Programas de computador I. Oliveira, Renato da Silva, 1960- II. Salvato, Carlos Eduardo Rocha, 1968- III. Série.

CDD-001.6425
-001.64
86-0999 -001.6424

Índices para catálogo sistemático:

1. BASIC : Linguagem de programação : Computadores : Processamento de dados 001.6424
2. Computadores : Programas : Processamento de dados 001.6425
3. MSX : Computadores : Processamento de dados 001.64
4. Programas : Computadores : Processamento de dados 001.6425
5. Programas aplicativos : Computadores : Processamento de dados 001.6425

SUMÁRIO

SUMÁRIO

NOTA DO EDITOR

Aprender BASIC é um pouco como aprender uma língua: não adianta nada ficar estudando horas e horas de gramática. O importante é a conversação!

Obviamente a gramática se torna importante numa segunda fase, depois que a competência linguística foi adquirida instintivamente.

No computador, o equivalente à conversação consiste na digitação de programas já prontos e na sua posterior adaptação através de um processo de tentativas, experiências e análise.

Neste segundo volume do COLEÇÃO DE PROGRAMAS PARA MSX, tentamos preencher esta necessidade.

O livro pode, então, ser usado em 3 diferentes níveis de profundidade. Numa primeira abordagem ele pode ser encarado como uma fonte de software impresso. O usuário principiante digita os programas e se limita a utilizá-los. Para este leitor, nos preocupamos com o tipo de programas incluídos neste volume: foi dada mais ênfase a aplicativos e utilitários, apesar de não termos esquecido os imprescindíveis jogos.

Como subproduto deste processo, o leitor acaba adquirindo cada vez mais familiaridade com a máquina e seu teclado e incorpora, de maneira instintiva, uma série de técnicas e truques de programação.

Num nível mais profundo, o leitor que já tem alguma fami-

liaridade com programação, estuda a análise que é feita após cada listagem para perceber as estruturas utilizadas e incorporar cada vez mais conhecimentos sobre o poderosíssimo BASIC MSX.

Finalmente, num nível mais profundo ainda, temos o leitor que se predispõe a alterar os programas introduzindo aperfeiçoamentos, adaptando-os às suas necessidades e personalizando-os.

Além de tentarmos atingir uma vasta gama de leitores com relação ao nível de familiarização com o BASIC e a máquina em si, tentamos também satisfazer um largo leque de usuários no que se refere ao campo de interesse.

Para o ADOLESCENTE temos os jogos de ação e os de inteligência. Para desfazer um pouco a imagem de "video game de luxo" que o MSX injustamente adquiriu, preferimos dar mais ênfase à inteligência, com os excelentes, entre outros, LABIRINTO e OTHELLO.

Nos jogos de ação, preferimos discutir detalhadamente a montagem do ISCAI JEGUE (paródia do divulgadíssimo SKY JAGAR) pois achamos mais importante desenvolver a habilidade criativa do adolescente, do que transformá-lo num simples "jogador".

Para o PROFISSIONAL que pretende utilizar seu MSX de uma maneira mais "séria", temos uma seção de aplicativos, dentre os quais podemos citar a MATEMÁTICA FINANCEIRA e o ESTATÍSTICA.

Para os PROFESSORES, preocupados em se familiarizar com esta nova e revolucionária ferramenta de ensino, temos programas didáticos que exemplificam algumas das quase infinitas aplicações do computador na escola.

Finalmente, ao PROGRAMADOR que deseja utilizar os recursos do MSX para produzir seu próprio software, dedicamos a seção de utilitários, onde oferecemos "programas-ferramentas", alguns dos quais foram desenvolvidos por nós para uso interno da editora.

Encerramos estas linhas com um alerta aos principiantes: diz um velho ditado que "os programadores, as virgens e os paraquedistas só podem errar uma vez!" Por isso, ao cometer algum erro, ao invés de "começar tudo de novo", gaste um pouco de seu tempo tentando descobrir onde e porque ele foi cometido. Assim diminui-se a possibilidade de cometê-lo de novo.

Dentre os erros mais comuns, existem os de digitação, principalmente na confusão dos símbolos: O e Ø ; 8 e B ;
1 e I ; 1 e l .

Bom trabalho!

ISCAI JEGUE

Num livro como este, as listagens de programas são tão importantes quanto o próprio texto. Assim sendo um dos cuidados que foram tomados foi o de se obter a listagem com uma impressora ligada ao computador, no qual o funcionamento do programa foi testado.

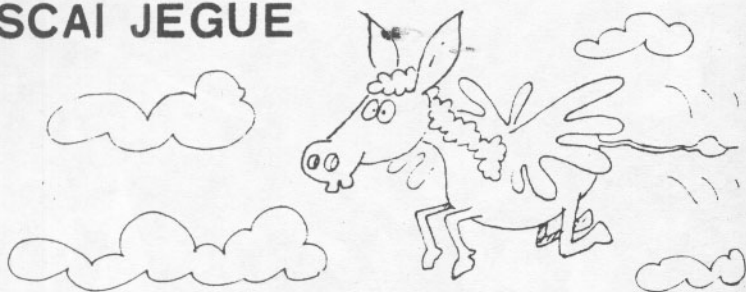
Desta forma reduzimos ao máximo a possibilidade de erros de transcrição. As listagens foram tiradas em 40 colunas, de maneira a assumirem o mesmo aspecto que têm na tela do MSX quando se usa a SCREEN 0, a mais utilizada para edição de programas. Isto facilita a contagem de caracteres repetidos e permite uma primeira "checagem" por parte do digitador, pois uma eventual diferença no alinhamento significa o esquecimento de algum caractere.

Para os que têm um HOTBIT ou EXPERT primeira versão, aconselhamos comandar

```
WIDTH 40
```

pois estes aparelhos, ao serem ligados, apresentam uma SCREEN 0 com 39 colunas.

Nesta primeira parte do livro, apresentamos um programa que deve ser digitado aos poucos, analisando cada trecho e tentando entender o porquê de cada comando e instrução. O leitor deve entender que, além de ser uma fonte muito barata de software, um livro como este representa, principalmente, um valioso instrumento de aprendizado.

ISCAI JEGUE

INSTRUÇÕES

Um dos recursos mais potentes do MSX para elaboração de jogos de ação, é a possibilidade de se utilizar a SCREEN 1 com caracteres redefinidos. O programa apresentado a seguir é uma caricatura de um dos mais belos jogos que usam um recurso semelhante a esse: o SKY JAGAR (ou COLUMBIA). Vamos estudar passo a passo os estágios para sua elaboração.

Antes de mais nada é necessário uma idéia. É a partir dela que desenvolveremos o jogo. Vamos usar a mesma idéia do SKY JAGAR, isto é, a tela (SCREEN 1) deverá "rolar" para baixo e o jogador deverá controlar um jegue (uma nave, no original), desviando-se dos obstáculos (buracos, mata-burros e cobras) e, se possível, destruindo-os.

O controle do jegue será feito através das setas ao lado direito do teclado. Para eliminar obstáculos à sua frente, o jóquei que montar o jegue deverá usar a barra de espaços.

Os demais detalhes do jogo irão surgindo naturalmente durante a sua elaboração.

Para começar, vamos introduzir e analisar a rotina que faz a tela girar na vertical, de cima para baixo (figura 1.1). Na verdade, não é toda a tela que gira. A última linha, na parte inferior do vídeo, fica imóvel, reservada para apresentação de mensagens permanentes. Você entenderá o porquê disso mais adiante.

Figura 1.1 — Giro vertical da SCREEN 1

```
1000 '  Rotina para "ROLAR" a
1010 '  SCREEN 1  para baixo:
1020 CLEAR 200,&HC000
1030 FOR I = &HC000 TO &HC039
```

```
1040        READ X$
1050        POKE I,VAL("&H"+X$)
1060 NEXT I
1070 DATA F3,21,BF,1A,11,DF,1A,0E
1080 DATA 16,06,20,7D,D3,99,7C,D3
1090 DATA 99,F5,F1,DB,98,F5,7B,D3
1100 DATA 99,7A,F6,40,D3,99,F1,00
1110 DATA D3,98,2B,1B,10,E5,0D,20
1120 DATA E0,97,D3,99,3E,58,D3,99
1130 DATA 06,20,78,D3,98,00,10,FB
1140 DATA FB,C9
1150 DEFUSR0=&HC000
```

A parte principal dessa rotina é um programa em linguagem de máquina, cujos códigos estão inseridos em hexadecimal nas linhas DATAs de 1070 e 1140.

A linha 1020 reserva 200 bytes para variáveis "strings" e fixa o maior endereço que o programa em BASIC poderá usar.

O laço entre as linhas 1030 e 1060 lê os códigos hexadecimais e os insere a partir do endereço &HC000 da RAM. O endereço de execução dessa rotina é definido pela linha 1150. Cada vez que a rotina em linguagem de máquina é executada, ela gira a SCREEN 1 uma linha para baixo (com exceção da última linha!).

Para testar essa rotina, acrescente ao programa as linhas mostradas na figura 1.2.

Figura 1.2 — Teste do Giro Vertical da SCREEN 1

```
1160 '    Teste da rotina
1170 COLOR 9,4,7 : SCREEN 1 : KEY OFF
1180 LOCATE 32*RND(1),0
1190 PRINT "x"
1200 K=USR0(0)
1210 GOTO 1180
```

A linha 1170 define as cores de frente, fundo e borda; seleciona a SCREEN 1 (pois é nela que a rotina em linguagem de máquina produz o movimento vertical) e apaga as mensagens das teclas de funções na parte inferior da tela.

A linha 1180 "sorteia" uma coluna na parte superior da tela e a linha 1190 imprime um "X" nessa posição.

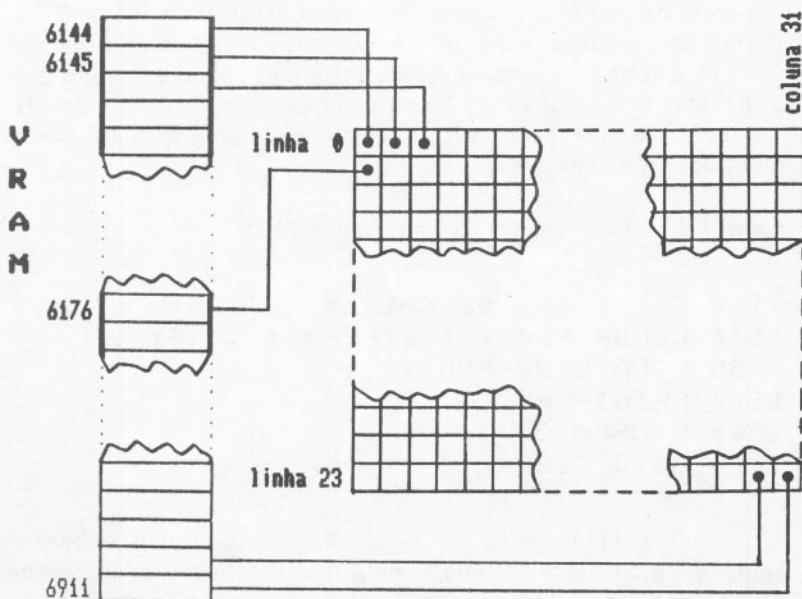
Logo a seguir, na linha 1200. a rotina em linguagem de máquina é executada através da instrução K=USR0(0), fazendo a tela toda "rolar" uma linha para baixo.

O comando GOTO na linha 1210 faz com que o processo seja repetido indefinidamente a partir da linha 1180.

Antes de prosseguirmos, vamos otimizar um pouco o que já fizemos.

A impressão de caracteres no vídeo pode ser feita por PRINT, como no teste da figura 1.2, ou por um VPOKE na VRAM. A segunda maneira é significativamente mais rápida que a primeira e, em jogos de ação, a velocidade é fundamental. Vamos, portanto, tentar usar o VPOKE ao invés do PRINT.

A imagem da SCREEN 1 é armazenada entre os endereços 6144 e 6911 (inclusive). O primeiro desses endereços (6144) contém o código do caractere apresentado no canto superior esquerdo da tela. Veja na figura 1.3 como a SCREEN 1 é mapeada e armazenada na VRAM.

Figura 1.3 — SCREEN 1

Portanto, para imprimir um caractere na linha superior da tela, devemos "vpokeá-lo" entre os endereços 6144 e 6144+31.

A sintaxe do VPOKE é:

VPOKE endereço, código do caractere

O código do caractere "x" é 120, portanto a instrução VPOKE 6144+32*RND(1),120 resolve o nosso problema.

Experimente eliminar as linhas 1188 e 1190 e substituí-las pela linha:

```
1180 VPOKE 6144+32*RND(1),120
```

Podemos usar o programa do jeito que ele está para produzir os obstáculos (representados pelo "x"). O carácter "x", entretanto, não se parece nem um pouco com um obstáculo. Vamos substituí-lo pelo caractere "Ω" cujo código é 234.

Substitua a linha 1180 por:

```
1180 VPOKE 6144+32*RND(1),234
```

Agora, já temos a tela girando para baixo com os obstáculos. Só falta o jegue correndo.

Provisoriamente, podemos usar o caractere "Δ" cujo código é 216, para representar o jegue.

Insira a linha:

```
1190 VPOKE 6640,216
```

O jegue já está na tela deixando atrás de si um rastro de sujeira, que posteriormente iremos eliminar.

Vamos agora dar-lhe movimento.

Elimine as linhas de 1180 a 1210 e insira as linhas mostradas na figura 1.4.

Figura 1.4 — Movimentando o jegue pela tela

```
1180 DEFINT A-Z:X=6640
1190 VPOKE 6144+32*RND(1),234
1200 ' Movimentacao do jegue
1210 C=STICK(0)
1220 IF C>=6 AND C<=8 AND (X-1) MOD 32 T
HEN X=X-1
1230 IF (C=1 OR C=2 OR C=8) AND X>6176 T
HEN X=X-32
```

```
1240 IF C>=2 AND C<=4 AND (X+1) MOD 32 T
HEN X=X+1
1250 IF C>=4 AND C<=6 AND X<6848 THEN X=
X+32
1260 K=USR0(0)
1270 VPOKE X,216
1280 GOTO 1190
```

A linha 1180 define todas as variáveis usadas no programa como inteiras (isso aumenta um pouco a velocidade) e atribui à variável X o valor 6640 (posição inicial do jegue na VRAM).

A linha 1190, como já sabemos, gera os obstáculos no topo da tela.

A linha 1210 verifica as teclas das setas à direita do teclado através da função STICK(0).

As linhas seguintes alteram o valor da variável X, de acordo com a tecla de seta pressionada. Para irmos para a esquerda, basta subtrair uma unidade de X. Para irmos para a direita, basta acrescentar uma unidade a X. Para subirmos uma linha na SCREEN 1, temos que subtrair 32 unidades de X. Para descermos uma linha, devemos fazer o contrário, isto é, somar 32 unidades ao valor de X.

Os operadores lógicos OR e AND são usados para permitir movimentos nas diagonais e a função MOD é usada para evitar que o jegue saia da tela pelas laterais, desaparecendo de um lado e surgindo do outro.

A rotina para girar a tela é executada através da linha 1260 e o jegue é impresso pela linha 1270.

A linha 1280 começa tudo de novo a partir da impressão dos obstáculos no topo da tela.

Com isso, temos o jegue com movimento e a tela girando. Atrás do jegue, porém, ainda forma-se um rastro que deve ser apagado. Isso pode ser feito alterando-se a linha 1210 para:

```
1210 VPOKE X,32 : C=STICK(0)
```

O caractere 32 corresponde ao espaço em branco. Ele é "vpokeado" sobre o jegue antes que a tela gire, eliminando assim a formação do rastro.

Nosso próximo passo será checar quando o jegue colide com um obstáculo (o caractere de código 234). Se isso acon-

tecer, devemos desviar o programa para uma rotina que trate especificamente desse caso. Vamos fazer essa checagem alterando ou acrescentando as linhas mostradas na figura 1.5.

Figura 1.5 — Checagem de choque com obstáculo

```
1270 IF VPEEK(X)=234 THEN 1310
1280 VPOKE X,216
1290 GOTO 1190
1300 ' Rotina para obstaculo
1310 PRINT"  O jegue caiu numa vala!!!"
1320 FOR F=1 TO 3000:NEXT F
1330 GOTO 1170
```

A linha 1270 verifica se o caractere logo acima do jegue na tela é um obstáculo (caractere 234) e, em caso afirmativo, desvia o programa para a rotina que inicia na linha 1310. Caso contrário, a linha 1280 é executada e o jegue (caractere 216) é impresso.

A linha 1290 retoma a execução a partir da impressão dos obstáculos no topo da tela.

A rotina que começa na linha 1310 apenas imprime uma mensagem no vídeo ("O jegue caiu numa vala!!!"), espera alguns segundos e desvia a execução para a linha 1170, onde a tela é limpa pelo comando SCREEN 1.

Até aqui estamos ignorando propositalmente um dos recursos mais atraentes dos MSX, a produção de sons.

Vamos inicialmente produzir um som para o deslocamento do jegue. Faremos isso de uma forma não muito usual (mais um macete!... preste atenção!).

Além do PLAY e do SOUND, que usam o PSG, pode-se produzir um click (o mesmo do teclado!) através de um OUT (comando do BASIC MSX) numa das portas da PPI.

Esse circuito controla, entre outras coisas, o click do teclado e você poderá obter mais detalhes sobre seu funcionamento no capítulo 3 do livro "APROFUNDANDO-SE NO MSX".

Mude a linha 1260 para:

```
1260 OUT 170,255:K=USR(0):OUT 170,127
```

Isso deverá dar som ao movimento do jegue.

Agora vamos melhorar a rotina do choque com o obstáculo. Altere o programa introduzindo as linhas mostradas na figura 1.6.

Figura 1.6 — Rotina de choque melhorada

```
1300 ' Rotina para obstaculo
1310 VPOKE X,42:SOUND 7,247:SOUND 6,31
1320 SOUND 8,16:SOUND 12,64:SOUND 13,0
1330 FORF=1TO50:IFF/2=F\2THENVPOKEX,35
1340 FOR G=1TO20:NEXTG:VPOKEX,42:NEXTF
1350 GOTO 1190
```

Nessa rotina, há duas coisas em que você deve prestar atenção: o ruído e a imagem do choque. Ambos foram produzidos de uma forma bastante simples e o resultado é plenamente satisfatório.

Agora vamos dar uma arma ao jegue, isto é, ao seu controlador, para que ele possa destruir os obstáculos. Faremos isso introduzindo uma rotina para lançar raios laser mediante pressão da barra de espaços.

Para ativar a interrupção do programa quando a barra for pressionada, devemos usar as instruções STRIG(0) ON e ON STRIG GOSUB. Altere a linha 1160 conforme segue:

```
1160 STRIG(0) ON:ON STRIG GOSUB 1370
```

Depois, introduzia a rotina apresentada na figura 1.7.

Figura 1.7 — Raio LASER disparado

```
1360 '  Raio laser disparado
1370 VPOKE X,216
1380 FOR F=X-32 TO X-512 STEP -32
1390 VPOKE F,33
1400 IF VPEEK(F-32)<>32 THEN 1420
1410 NEXT F
1420 FOR G=X-32 TO F-32 STEP-32
1430 VPOKE G,32
1440 NEXT G
1450 VPOKE X,32
1460 RETURN
```

A linha 1370 coloca o jegue na tela.

O laço entre as linhas 1380 e 1410 lança em raio formado por uma sequência de caracteres "!" (código 33) a partir do jegue. Dentro do laço, a linha 1400 verifica se algum obstáculo foi atingido e, em caso positivo, desvia o programa para a linha 1420.

O laço entre as linhas 1420 e 1450 apenas apaga o raio lançado.

Finalmente, a linha 1460 retorna para o programa principal.

Com essa nova arma, o jegue ficou invencível! Um SUPER-ISCAI-JEGUE! O jogo, desse jeito, não tem graça. Vamos melhorar um pouco essa rotina, fazendo com que, ao errar um tiro, o disparador laser fique algum tempo sem funcionar.

Introduza e altere algumas linhas do programa conforme segue:

```
1200 CT=CT+1:IF CT=10 THEN FL=1:CT=0
1370 IFFL=0THENRETURNELSEVPOKEX,216
1410 NEXT F : FL=0 : CT=0
```

Essas linhas fazem com que o raio laser fique algum tempo desativado após um disparo ser dado sem atingir nenhum obstáculo.

Ainda assim, o jogo está um pouco sem graça, pois não há limite para a quantidade de colisões que o jegue pode ter com os obstáculos. Vamos limitar esse número em três, acrescentando ao programa as alterações mostradas na figura 1.8.

Figura 1.8 — Finalização do jogo

```
1180 DEFINT A-Z:X=6640:N=3
1190 VPOKE 6144+32*RND(1),234:LOCATE 0,2
3:PRINT"CHANCES:";STRING$(N,216);" ";
1350 N=N-1:IF N>0 THEN 1190 ELSE 1470
1470 ' rotina de finalizacao
1480 COLOR 14,1,13 : SCREEN 1
1490 PRINT" O JEGUE MORREU !!!"
1500 PRINT" Para obter outro JEGUE"
1510 PRINT" digite a tecla RETURN."
1520 INPUT X$
1530 RUN
```

Agora, podemos dizer que já temos um jogo pronto. A partir daqui podemos começar a enfeitá-lo.

Vamos fazer com que a cada obstáculo destruído o jegue ganhe um ponto e, a cada 100 pontos acumulados, faremos com que ele ganhe uma nova chance. Para isso basta introduzir as alterações apresentadas na figura 1.9.

Figura 1.9 — Contagem de pontos e bônus

```
1180 DEFINT A-Z:X=6640:N=3:PT=0
1190 VPOKE 6144+32*RND(1),234:LOCATE 0,2
3:PRINT"CHANCES:";STRING$(N,216);STRING$
(4-N,32);" PONTOS:";USING"###";PT;
1270 IF VPEEK(X)=234 THEN 1300
1300 PT=0
1400 IF VPEEK(F-32)<>32 THEN PT=PT+1:GOT
O 1420
1440 NEXT G:IF PT=100 THEN PT=0:N=N+1:IF
 N=5 THEN N=4
```

Mesmo com todos os caracteres impressos, a tela do jogo ainda está um pouco pobre, pois só tem três cores. Na SCREEN 1 existe a possibilidade de definirmos as cores de cada grupo de 8 caracteres seguidos. O MSX dispõe de 256 caracteres numerados de 0 a 255. Portanto, temos 32 grupos de 8 caracteres cada. O primeiro grupo começa com o caractere de código 0 e termina com o caractere de código 7. O último grupo é o dos caracteres com códigos entre 248 e 255. Para cada grupo, podemos definir uma cor de frente e uma cor de fundo diferentes.

A região da VRAM que armazena as cores de frente e fundo desses 32 grupos de caracteres fica entre os endereços 8192 e 8223. Na tabela da figura 1.10 podemos observar os trinta e dois "octetos" de caracteres e os respectivos endereços na VRAM onde suas cores de frente e fundo estão definidas.

Vamos alterar a cor do jegue. O código de seu caractere é 216, portanto o endereço da VRAM onde suas cores são definidas é o 8219.

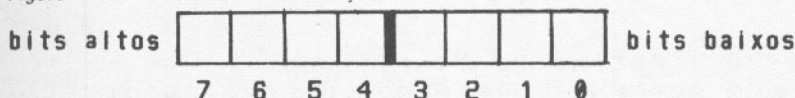
O byte 8219 da VRAM, como qualquer outro, possui 8 bits. Cada bit pode valer 0 ou 1 (figura 1.11).

Os 4 bits mais baixos desse byte (os da direita) devem armazenar a cor de fundo e os 4 bits mais altos (os da esquerda) devem armazenar a cor de frente. Note que cada grupo de 4 bits pode armazenar um número entre 0 e 15 (em binário). Por exemplo, se inserirmos o número binário &B10110100 no endereço 8219 da VRAM o jegue será desenhado em amarelo sobre fundo azul, pois &B1011 = 11 (cor amarela) corresponde

à cor de frente e &B0100=4 (cor azul) representa a cor de fundo!

Figura 1.10 — Tabela de cores da SCREEN 1

Endereço	Caracteres alterados							
8192	0	1	2	3	4	5	6	7
8193	8	9	10	11	12	13	14	15
8194	16	17	18	19	20	21	22	23
8195	24	25	26	27	28	29	30	31
8196	32	33	34	35	36	37	38	39
8197	40	41	42	43	44	45	46	47
8198	48	49	50	51	52	53	54	55
8199	56	57	58	59	60	61	62	63
8200	64	65	66	67	68	69	70	71
8201	72	73	74	75	76	77	78	79
8202	80	81	82	83	84	85	86	87
8203	88	89	90	91	92	93	94	95
8204	96	97	98	99	100	101	102	103
8205	104	105	106	107	108	109	110	111
8206	112	113	114	115	116	117	118	119
8207	120	121	122	123	124	125	126	127
8208	128	129	130	131	132	133	134	135
8209	136	137	138	139	140	141	142	143
8210	144	145	146	147	148	149	150	151
8211	152	153	154	155	156	157	158	159
8212	160	161	162	163	164	165	166	167
8213	168	169	170	171	172	173	174	175
8214	176	177	178	179	180	181	182	183
8215	184	185	186	187	188	189	190	191
8216	192	193	194	195	196	197	198	199
8217	200	201	202	203	204	205	206	207
8218	208	209	210	211	212	213	214	215
8219	216	217	218	219	220	221	222	223
8220	224	225	226	227	228	229	230	231
8221	232	233	234	235	236	237	238	239
8222	240	241	242	243	244	245	246	247
8223	248	249	250	251	252	253	254	255

Figura 1.11 — Estrutura de um byte

Introduza a linha a seguir no programa.

```
1175 VPOKE 8219,&B10110100
```

Ela servirá para destacarmos o jegue dos demais caracteres impressos.

Aproveite para melhorar um pouco a linha 1170, deixando-a como segue:

```
1170 COLOR9,4,7:SCREEN1,,0:KEYOFF
```

Dessa forma eliminamos o click de teclado.

Vamos agora colorir mais ainda o jogo. Inicialmente podemos mudar a cor do raio laser de vermelho para cinza (cor 14=&B1110). O caractere usado para produzir o raio é o "!" cujo código é 33. Olhando a tabela da figura 1.10 vemos que o endereço a ser alterado é o 8196, portanto basta introduzir a linha 1176 mostrada a seguir para fazê-lo ficar cinza e azul.

```
1176 VPOKE 8196,&B11100100
```

Agora vamos colorir todas as mensagens mostradas na parte inferior da tela, deixando-as brancas. Os caracteres usados são as letras A,C,E,H,N,A,P,S,T e o sinal ":". Introduza a linha a seguir:

```
1177 FOR F=8198 TO 8202 : VPOKE F,&B1111
0100 : NEXT F
```

Uma outra implementação que podemos fazer é produzir um som toda vez que o raio laser for disparado. A linha adiante se encarrega disso.

```
1375 SOUND7,52:SOUND6,8:SOUND8,16:SOUND9
,16:SOUND0,99:SOUND1,0:SOUND2,199:SOUND3
,0:SOUND12,3:SOUND13,4
```

Agora vamos mexer mais um pouco no final do jogo. Tocaremos uma marcha fúnebre para o jegue morto. Na figura 1.12 vemos as linhas a serem acrescidas à rotina de finalização.

Figura 1.12 — A Morte do Jegue com Pompas

```
1470 ' rotina de finalizacao
1480 SCREEN1:FORF=0TO99:COLOR0,FMOD15
```

```
1490 NEXTF:COLOR6,1,13:SOUND7,56
1495 LOCATE5,5:PRINT"O JEGUE MORREU !"
1500 PRINT" Para ressuscitar o JEGUE"
1510 PRINT"digite a barra de espacos."
1520 PLAY"t110m100s0v15","t110v15"
1530 PLAY"o1bbbbo2dc#c#o1bba#br8","o3bbb
bo4dc#c#o3bba#br8"
1540 STRIG(0)OFF
1550 IFSTRIG(0)=0THEN1550ELSERUN
```

Como você deve estar notando, podemos fazer mil coisas para melhorar cada vez mais o jogo. Antes de deixarmos as alterações por sua conta, vamos fazer mais uma: redefiniremos os caracteres do jegue e do obstáculo!

Assim que a instrução SCREEN 1 é executada, o MSX copia todos os 256 caracteres da ROM para uma região da VRAM entre os endereços 0 e 2047. Cada caractere é representado por oito bytes em sequência (figura 1.13).

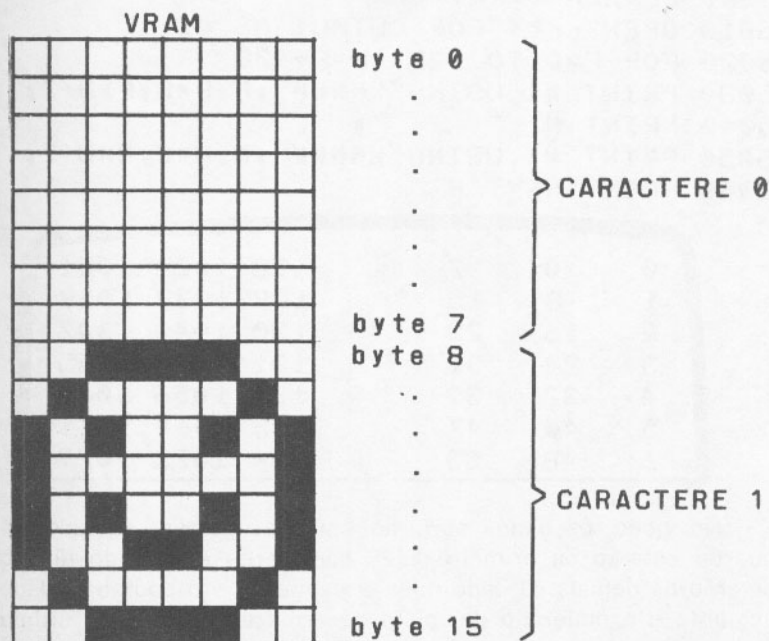
Figura 1.13 — Caracteres na VRAM

Por exemplo, o caractere "A" é armazenado como mostra a figura 1.14.

Figura 1.14 — O caractere "A" na VRAM

```
1728 ----> 00000000 ---->
1729 ----> 00100000 ---->
1730 ----> 00100000 ---->
1731 ----> 01010000 ---->
1732 ----> 01010000 ---->
1733 ----> 10001000 ---->
1734 ----> 11111000 ---->
1735 ----> 00000000 ---->
```

O programa da figura 1.15 apresenta no vídeo os endereços iniciais e finais de armazenamento dos 256 caracteres na VRAM quando se usa a SCREEN 1.

Figura 1.15 — Endereços de caracteres na VRAM

```
5000 SCREEN 0:KEY OFF
5010 OPEN"crt:"FOR OUTPUT AS #1
5020 FOR F=0 TO 127 :H=F+128
5030 PRINT #1,USING"#####";F;F*8;F*8+7;
5040 PRINT #1,"     ";
5050 PRINT #1,USING"#####";H;H*8;H*8+7;
5060 PRINT:NEXT F
```

```
0    0    7         128 1024 1031
1    8   15         129 1032 1039
2   16   23         130 1040 1047
3   24   31         131 1048 1055
4   32   39         132 1056 1063
5   40   47         133 1064 1071
6   48   55         134 1072 1079
```

No vídeo, os dados surgirão em duas partes: ao lado esquerdo estarão os primeiros 128 caracteres e ao lado direito estarão os demais. O dado mais à esquerda é o código, o dado seguinte é o endereço de início de armazenamento e o último dado é o endereço final de armazenamento.

Para redefinir um caractere basta alterar os bytes que o definem na VRAM.

Vamos alterar o jegue (código 216) e os obstáculos (código 234). Usando o programa da figura 1.15 vemos que eles são armazenados, respectivamente, nos endereços 1728-1735 e 1872-1879. Vamos alterá-los, deixando-os como na figura 1.16.

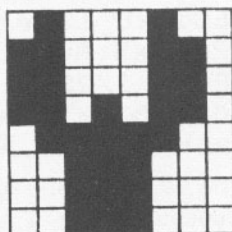
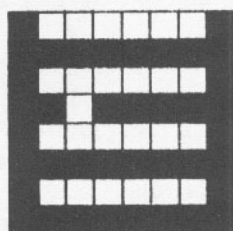
Figura 1.16 — O jegue e os obstáculos redefinidos

Introduza as linhas mostradas na figura 1.17 para redefinir os caracteres.

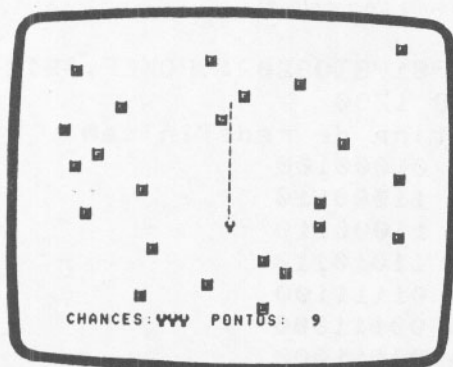
Figura 1.17 — Rotina de redefinição

```
1177 FORF=8198TO8202:VPOKEF,&B11110100:N
EXTF:GOSUB 1730
1560 ' Rotina de redefinicao
1570 DATA 01000100
1580 DATA 11000110
1590 DATA 11000110
1600 DATA 11010110
1610 DATA 01111100
1620 DATA 00111000
1630 DATA 00111000
```

```
1640 DATA 00111000
1650 DATA 10000001
1660 DATA 11111111
1670 DATA 10000001
1680 DATA 11011111
1690 DATA 10000001
1700 DATA 11111111
1710 DATA 10000001
1720 DATA 11111111
1730 RESTORE 1570:FOR F=0 TO 7
1740 READ X$:A=VAL("&b"+X$)
1750 VPOKE 1728+F,A : NEXT F
1760 FORF=0TO7:READX$:A=VAL("&B"+X$)
1770 VPOKE1872+F,A:NEXTF:RETURN
```

A partir daqui as alterações ficam por sua conta. Aqui vão algumas sugestões:

0) Crie uma tela de apresentação para o jogo!
1) Redefina os caracteres "espaço em branco" cujo código é 32.
2) Redefina o caractere "!", cujo código é 33.
3) Gere ao menos 3 tipos diferentes de obstáculos (não apenas o caractere 234) e faça com que um deles tenha que ser necessariamente destruído antes de chegar ao fim da tela.
4) Transforme o JEGUE num jato e os obstáculos em naves inimigas.
5) Não esqueça de gravar o programa !!!"
6) Jogue !!!

No livro "Programação Avançada em MSX" ensinamos outros truques que se pode fazer com as telas no MSX.

JOGOS DE AÇÃO

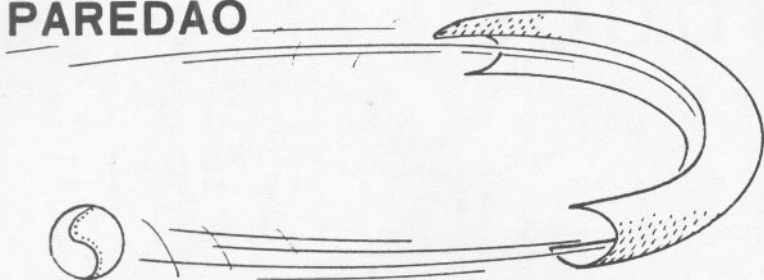
Quando o padrão MSX foi lançado no Japão, a estratégia de "marketing" utilizada para convencer os compradores a colocar um micro dentro de casa, foi a de apresentá-lo como um sofisticado video-game, brinquedo então em auge na preferência dos consumidores.

Isto explica a abundância de fitas e cartuchos gravados com jogos de ação que até agora se encontram no mercado. Na realidade foi cometida uma injustiça com esta máquina maravilhosa que tem possibilidade de uso muito mais complexa do que a de rodar simples joguinhos de 'flipperama".

Ao colocarmos esta seção de jogos de ação no livro, nossa intenção não foi a de simplesmente fornecer ao leitor mais programas deste tipo, pois os que já existem no mercado são mais que suficientes, tanto em qualidade quanto em quantidade, para satisfazer o aficcionado mais exigente.

Nosso verdadeiro objetivo foi o de motivar o leitor a entender como estes programas são elaborados, incentivando-o a introduzir suas próprias alterações, melhorando e personalizando o software sugerido.

A grande diferença entre um micro computador e um video-game não está na máquina em si, mas sim na atitude mental de quem a opera. No micro, o usuário tem condições de desenvolver sua própria criatividade, sendo este o resultado verdadeiramente importante.

PAREDÃO

INSTRUÇÕES

Este programa é uma versão em BASIC de um dos primeiros "video-games" produzidos comercialmente. Obviamente, a velocidade do jogo é bastante lenta quando o comparamos com versões feitas em linguagem de máquina, no entanto, a estrutura do programa é o que mais nos importa no momento e compreendê-la poderá ser de grande utilidade quando você for desenvolver novos programas.

O jogo é bastante simples: você controla uma raquete com as teclas ▲ e ▼ e deve impedir que uma bola de tênis saia pelo lado direito da tela (quadra). Há um total de 6 bolas, ao término das quais o jogo acaba.

Figura 2.1 — Programa PAREDÃO

```
1000 REM  JOGO PAREDAO
1010 REM  Rubens & Renato (FEV/86)
1020 REM  ------------------------------
1030 CQ=3 : CP=15 : CR=6
1040 CB=9 : CN=12
1050 COLOR CP,CQ,CQ
1060 SCREEN 3,,0
1070 OPEN "GRP:" FOR OUTPUT AS #1
1080 FOR F=1 TO 5
1090   READ A$
1100   B$=B$+CHR$(VAL("&h"+A$))
1110 NEXT F
1120 DATA 70,c8,e8,f8,70
1130 SPRITE$(1)=B$
1140 DRAW"BM255,0L255D191R255"
1150 COLOR CR
```

```
1160 PSET(230,80),CR : DRAW"D16"
1170 XR=230 : YR=80
1180 PT=0
1190 FOR NB=6 TO 1 STEP -1
1200  XB=8 : YB=INT(180*RND(-TIME))+4
1210  DX=4 : DY=RND(-TIME)*9-4
1220  COLOR CQ
1230  PSET (40,40),CQ : PRINT #1,NB+1
1240  COLOR CN
1250  PSET (40,40),CQ : PRINT #1,NB
1260  IF STICK(0)=1 AND YR>7 THEN PSET(X
R,YR+16),CQ : YR=YR-4 : PSET(XR,YR),CR
1270  IF STICK(0)= 5 AND YR<168 THEN PSE
T(XR,YR),CQ : YR=YR+4 : PSET(XR,YR+16),C
R
1280  XB=XB+DX : YB=YB+DY
1290  IF XB>260 THEN 1510
1300  IF XB<4 THEN DX=-DX : XB=XB+DX
1310  IF YB<4 OR YB>183 THEN DY=-DY:YB=Y
B+DY
1320  IF XB>XR-4 AND XB<XR+1 AND YB>YR-4
 AND YB<YR+16 THEN 1350
1330  PUT SPRITE 1,(XB,YB),CB,1
1340  GOTO 1260
1350  SOUND 0,250
1360  SOUND 1,1
1370  SOUND 2,250
1380  SOUND 3,0
1390  SOUND 7,56
1400  SOUND 8,16
1410  SOUND 9,16
1420  SOUND 11,255
1430  SOUND 12,2
1440  SOUND 13,0
1450  IF YB<YR+4 OR YB>YR+16 THEN NX=4 :
 NY=4 : GOTO 1480
1460  IF YB<YR+8 OR YB>YR+12 THEN NX=4 :
 NY=2 : GOTO 1480
1470  NX=4 : NY=1
1480  DX=-NX : DY=SGN(DY)*NY
1490  PT=PT+1
```

```
1500  GOTO 1280
1510 NEXT NB
1520 LINE(4,4)-(251,187),4,BF
1530 PSET(20,20),4 : PRINT #1,"PONTOS: "
;PT
1540 IF STRIG(0)=0 THEN 1540
1550 CLOSE : RESTORE : RUN
```

ANÁLISE

Os nomes das variáveis indicam as funções a que elas se prestam.

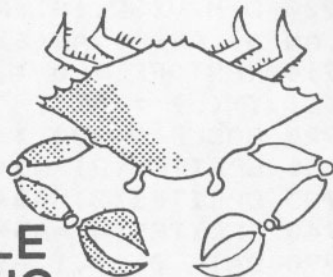
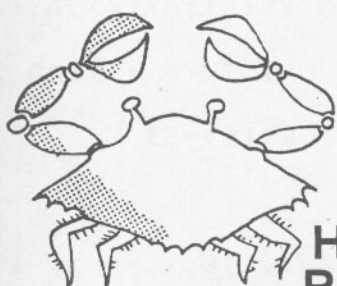
CQ = cor da quadra
CP = cor dos pontos
CB = cor da bola
CN = cor dos números
PT = pontos
NB = número de bolas
XB = coordenada X da bola
YB = coordenada Y da bola
DX = incremento (Delta) na coordenada X da bola
DY = incremento (Delta) na coordenada Y da bola
XR = coordenada X da raquete
YR = coordenada Y da raquete
NX = número do incremento de X da bola
NY = número do incremento de Y da bola

A bola é um "sprite" definido entre as linhas 1110 e 1170.

As linhas 1300 e 1310 movem a raquete.

Para iniciar o jogo, pressione a barra de espaços.

Um exercício interessante seria tentar fazer este mesmo jogo usando a SCREEN 1 ao invés da SCREEN 3 e nesse caso, certamente a velocidade do jogo seria bem maior!

Se você está disposto, aqui fica o desafio!

INSTRUÇÕES

Uma cidade foi invadida por terríveis siris, sendo que estes se refugiaram num edifício em construção.

Seu objetivo é destruir os invasores nesse edifício, cavando buracos no chão e fazendo-os cair.

Mova-se com as setas e pressione a barra de espaços para cavar buracos. Você tem cinco vidas para completar sua missão e perderá uma cada vez que um siri o atacar. Os buracos só podem ser feitos a partir do primeiro andar, sendo impossível fazê-los no térreo. O buraco será escavado sempre ao seu lado e na direção apontada pelas setas. Se você cair num dos buracos, nada lhe acontecerá.

A cada leva de siris que for destruída, um novo prédio será gerado, até que você perca todas as suas vidas e seja devorado pelos siris!

Tome cuidado ao digitar as linhas 900, 30000, 30010, 30020 e 30030, pois elas contém muitos dados numéricos importantes para a formação dos desenhos.

Nas linhas entre a 1220 e a 1275, preste atenção para não errar o número de espaços em branco, para que as mensagens saiam centralizadas.

As notas usadas pelo comando PLAY na linha 9010 devem ser digitadas com cuidado.

Figura 3.1 — Programa HOLE PANIC

```
10 REM        HOLE PANIC
20 REM        Autores: Luiz Fernando
30 REM                 Wilson  e  Yeh
100 SCREEN 2,2 : CLS
101 OPEN "grp:" FOR OUTPUT AS # 1
110 R=RND(1)
200 DEFINT A-Z
```

```
900 DIM V(32,10),XX(8),YY(8),AX(6),AY(6)
,AM(6),AB(6),AF(6)
910 RESTORE 990 : FOR L=1 TO 8 : READ XX
(L),YY(L) : NEXT
920 GOSUB 30000 : SPRITE$(1)=E$
930 SPRITE$(2)=A$+B$
935 SPRITE$(3)=A$+B$
940 SPRITE$(4)=A$+B$
990 DATA 0,-1,1,-1,1,0,1,1,0,1,-1,1,-1,0
,-1,1
1010 '
1200 SC=0:LE=5
1210 COLOR 15,5,7:CLS:PSET(0,50),4
1220 PRINT#1,"                P L A Y
1230 PRINT#1,:COLOR 6:PRINT#1,"
 Hole-Panic!!":PRINT#1,
1260 PRINT#1,:COLOR 7
1270 PRINT#1," escolha Joy-Stick ou Tec
lado"
1275 PRINT#1,"              (J) ou (T)"
1280 A$=INKEY$:IFA$=""THENR=RND(1):GOTO
1280
1290 IFA$="t"ORA$="T"THEN2000
1300 IFA$="j"ORA$="J"THENM=M+1:GOTO2000
1310 GOTO1280
2000 FORL=0TO10:FORK=0TO31:V(K,L)=0:NEXT
:NEXT
2001 COLOR10,1,1:CLS:FORL=0TO3
2010 LINE(0,L*48+32)-(256,L*48+39),11,BF
2020 FORK=0TO31:V(K,L*3)=1:NEXT:NEXT
2030 FORL=0TO2
2040 FORK=0TO2
2050 R=RND(1)*31
2060 V(R,L*3+1)=1:V(R,L*3+2)=1
2070 LINE(R*8+2,L*48+40)-(R*8+2,L*48+80)
,11:LINE(R*8+13,L*48+40)-(R*8+13,L*48+80
),11
2080 FORJ=4TO46STEP6:LINE(R*8+2,L*48+40+
J)-(R*8+13,L*48+40+J),12:NEXT
2090 NEXT:NEXT
2200 X=15:Y=9
```

```
2300 COLOR 4:PSET(0,0),2:PRINT#1,"   Hol
e Panic:Score:":COLOR 15
2400 AK=0:FORL=2TO4:AH(L)=1:AY(L)=(L-2)*
3:AX(L)=RND(1)*29+1:AF(L)=1:NEXT
3000 V(X,Y)=BF:S=STICK(M):KX=X+XX(S):KY=
Y+YY(S):IFKX<0ORKX>31ORKY<0ORKY>10THEN30
50
3010 IFV(KX,KY)=3THENV(KX,KY)=1:LINE(X*8
-8,Y*16+32)-(X*8+23,Y*16+39),11,BF:X=KX:
Y=KY+3:PUTSPRITE1,(X*8,Y*16+16),4:GOTO 3
050
3015 IFV(KX,KY)<>0THENX=KX:Y=KY:PUTSPRIT
E1,(X*8,Y*16+16),4
3050 IFXX(S)<>0THENX1=XX(S)
3060 BF=V(X,Y):V(X,Y)=4
3100 FORL=2TO4:IFAF(L)=0THEN3900
3101 IFAM(L)<>0THEN3500
3104 IFAH(L)=0THENAH(L)=1
3105 IFAY(L)=YTHEN3140
3106 IFAY(L)<>0THENP1=V(AX(L),AY(L)-1)EL
SEP1=0
3110 IFAY(L)<>9THENP2=V(AX(L),AY(L)+1)EL
SEP2=0
3120 IFY>AY(L)ANDP2=1THENAM(L)=2:GOTO 35
00
3130 IFY<AY(L)ANDP1=1THENAM(L)=1:GOTO 35
00
3135 IFINT(RND(1)*20)=0THENAH(L)=-AH(L)
3140 KX=AX(L)+AH(L):IFKX<0ORKX>31THENAH(
L)=-AH(L):GOTO3140
3150 AX(L)=KX:F=V(AX(L),AY(L)):IFF=3THEN
GOSUB5000:GOTO 3900ELSEIFF=4THENOF=1
3151 AX(L)=KX:PUTSPRITEL,(AX(L)*8,AY(L)*
16+16),8:GOTO 3900
3500 AY(L)=AY(L)+AM(L)*2-3:A=AY(L):IFA=0
ORA=3ORA=6ORA=9THENAM(L)=0:AH(L)=SGN(X-A
X(L)):GOTO3105
3510 PUTSPRITEL,(AX(L)*8,AY(L)*16+16),8
3900 NEXT
4000 IFSTRIG(M)=0THEN4300
4005 IFY=9THEN4300
```

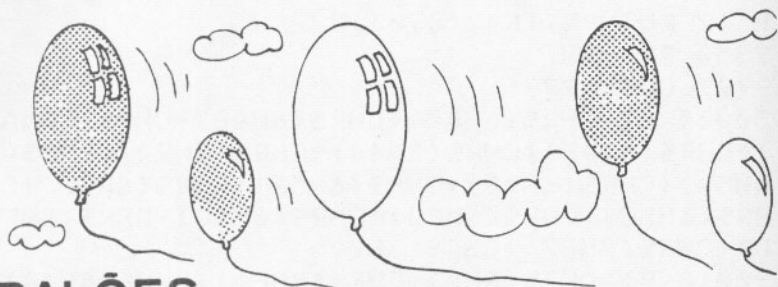
```
4006 IFX+X1*3<1ORX+X1*3>30THEN4300ELSEX2
=X*8+X1*16
4010 IFV(X+X1*2,Y+1)=1THEN4300
4011 IFY<>0THENIFV(X+X1*2,Y-1)=1THEN4300
4020 LINE(X2,Y*16+32)-(X2+15,Y*16+39),1,
BF:V(X+X1*2,Y)=3
4300 '
4400 IFOF=1THEN7000
4410 OF=0
4990 GOTO 3000
5000 PLAY"a64b64":AF(L)=0:PUTSPRITEL,(0,
0),2
5010 COLOR15,1:LINE(155,0)-(220,9),1,BF:
SC=SC+100:PSET(155,0),7:PRINT#1,SC;:LINE
(AX(L)*8-8,AY(L)*16+32)-(AX(L)*8+23,AY(L
)*16+39),11,BF:V(AX(L),AY(L))=1
5020 AK=AK+1:IFAK=3THEN9000
5090 RETURN
7000 OF=0:LE=LE-1:PLAY"o4eco3a":GOSUB950
0::IFLE=0THEN7200
7010 GOTO2000
7200 PSET(100,100),2:PRINT#1,"GAME OVER"
7210 FORT=1TO1000:NEXTT:CLS:COLOR4
7220 PRINT#1,"Deseja jogar novamente (s/
n)"
7225 A$=INKEY$:IFA$=""THEN7225
7227 IFA$="N"ORA$="n"THENEND
7230 IFA$="S"ORA$="s"THENLE=5:GOTO8000
7240 GOTO7225
8000 PUTSPRITE1,(0,0),2
8010 PUTSPRITE2,(0,0),2
8020 PUTSPRITE3,(0,0),2
8032 PUTSPRITE4,(0,0),2
8090 GOTO2000
9000 PSET(100,100),2:PRINT#1,"Clear!!"
9010 PLAY"o4ce8fg8abo5c8d8c":GOSUB9500
9020 GOTO2000
9500 IFPLAY(0)=-1THEN9500
9510 PUTSPRITE1,(0,0),2
9520 PUTSPRITE2,(0,0),2
9530 PUTSPRITE3,(0,0),2
```

```
9542 PUTSPRITE4,(0,0),2
9550 RETURN
9999 GOTO9999
30000 A$=CHR$(&HA0)+CHR$(&HAE)+CHR$(&HAA
)+CHR$(&HAE)+CHR$(&H44)+CHR$(&H24)+CHR$(
&H54)+CHR$(&HAF)+CHR$(&H5F)+CHR$(&HBF)+C
HR$(&H5C)+CHR$(&HBD)+CHR$(&H1C)+CHR$(&HF
D)+CHR$(&H87)+CHR$(&H0)
30010 B$=CHR$(&HA)+CHR$(&HEA)+CHR$(&HAA)
+CHR$(&HEA)+CHR$(&H44)+CHR$(&H48)+CHR$(&
H52)+CHR$(&HE5)+CHR$(&HFA)+CHR$(&HFD)+CH
R$(&H3A)+CHR$(&HBD)+CHR$(&H38)+CHR$(&HBF
)+CHR$(&HE1)
30020 DATA 3,7,7,7,3,1,15,31,27,19,7,14,
14,14,14,30
30030 DATA 192,224,224,224,192,128,240,2
48,216,200,112,112,112,112,112,120
30040 RESTORE 30020:E$=""
30050 FORL=0TO31:READS:E$=E$+CHR$(S):NEX
T
30060 RETURN
```

ANÁLISE

Nas linhas de 100 a 990 é realizada a inicialização do programa, dimensionamento de variáveis e definição dos "sprites". Da linha 1010 a 1270 temos a abertura. A seleção de teclado ou "joystick" é feita entre as linhas 1280 e 1310. Os pisos dos andares são montados nas linhas de 2000 a 2020. As escadas são geradas nas linhas de 2030 a 2090. Os siris são posicionados inicialmente pela linha 2400.

Na linha 3000 a função STICK identifica uma tecla pressionada. A linha 3010 faz a impressão do homem. Seu movimento é feito pela linha 3015. As linhas de 3100 a 3900 fazem comparações para checar as posições dos siris e se há algo em sua proximidade. Os siris são impressos pelas linhas 3510 e 3151. Nas linhas de 4000 a 4020 são feitas comparações para saber se a barra de espaços ou o disparador do "joystick" foi pressionado. As linhas 7000 e 7010 diminuem suas vidas e checam quantas ainda restam.

A partir da linha 30000 está a rotina de inicialização, que é chamada por um GOSUB na linha 920.

BALÕES

INSTRUÇÕES

Você odeia balões de aniversário. Eles perturbam sua mente e o deixam nervoso. Você precisa destruí-los a qualquer custo! Nem que tenha que gastar todo o seu tempo disponível para isso.

Parece o pensamento de uma mente doentia, mas na verdade é o tema do jogo descrito a seguir.

Seu objetivo é estourar os balões que ficam voando pela tela, encurralando-os com blocos de tijolos. Você controla os blocos com as setas ao lado direito do teclado e sempre que um balão ficar encurralado, ele estoura.

Tome cuidado na digitação das linhas entre 100 e 200, pois esses dados são facilmente alternáveis.

As linhas entre 1030 e 1060 também são perigosas. Digite-as com bastante atenção.

Figura 4.1 — Programa BALÕES

```
10 REM      BALOES
20 REM      Autores: Luiz Fernando
30 REM               Wilson  e  Yeh
100 DATA00,77,77,77,00,dd,dd,dd,00,77,77
,77,00,dd,dd,dd
110 DATA07,1f,3f,7f,67,db,bd,ff,ff,ff,ef
,77,78,3f,1f,07
120 DATAe0,f8,fc,fe,e6,db,bd,ff,ff,ff,f7
,ee,1e,fc,f8,e0
130 DATA0c,1e,3f,5d,fb,ff,c1,9c,af,32,3d
,3b,37,0f,07,03
140 DATA30,78,fc,ee,e3,ff,83,39,f5,4c,bc
,dc,ec,f0,e0,c0
150 DATA6b,6b,6b,6b,6b,6b,6b,6b,6b,6a,6a
,6a,6a,6a,6a,6a,6a
```

```
160 DATA21,21,21,21,21,21,21,21,21,21,21
,21,21,21,21,21
170 DATA21,21,21,21,21,21,21,21,21,21,21
,21,21,21,21,21
180 DATAb1,b1,b1,b1,b1,b1,b1,b1,b1,b1,b1
,b1,b1,b1,b1,b1
190 DATAb1,b1,b1,b1,b1,b1,b1,b1,b1,b1,b1
,b1,b1,b1,b1,b1
200 DATA32,-1,-1,32,1
210 ST=1000:A=RND(-TIME):GOSUB1250
220 FORI=0TO79:READA$:A=VAL("&h"+A$)
230 VPOKEI,A:VPOKEI+2048,A:VPOKEI+4096,A
:NEXT
240 FORI=0TO79:READA$:A=VAL("&h"+A$)
250 VPOKEI+8192,A:VPOKEI+10240,A:VPOKEI+
12288,A:NEXT
260 FORI=0TO11:VPOKEI+6144,12:VPOKEI+640
0,12:VPOKEI+6656,12:NEXT
270 DIMFA!(4),F%(4),BL!(6)
280 HS=500
290 SC=0
300 FORI=0TO4:FA!(I)=&H1863+I*2:NEXT
310 TI=ST:MA=6544:FK=5
320 GOSUB1240:FORI=2TO31:VPOKE&H1840+I,1
:VPOKE&H1AE0+I,1:NEXT
330 FORI=3TO22:VPOKEI*32+&H1802,1:VPOKEI
*32+&H181F,1:NEXT
340 VPOKEMA,6:VPOKEMA+1,8:VPOKEMA+32,7:V
POKEMA+33,9
350 FORI=0TO4:VPOKEFA!(I),2:VPOKEFA!(I)+
1,4:VPOKEFA!(I)+32,3
360 VPOKEFA!(I)+33,5:NEXT
370 FORI=0TO6
380 X=INT(RND(1)*27)+3:Y=INT(RND(1)*16)+
7
390 FORJ=0TO1:FORK=0TO1:IFVPEEK(&H1800+X
+Y*32+J+K*32)<>12GOTO 380
400 NEXTK,J
410 A=&H1800+X+Y*32:BL!(I)=A:VPOKEA,0:VP
OKEA+1,0:VPOKEA+32,0:VPOKEA+33,0
420 NEXT
```

```
430 GOSUB1190
435 GOSUB1210
440 RESTORE200:FORI=0TO4:READF%(I):NEXT
450 A=STICK(0):IFA=0GOTO 700
460 IFA=1THENP=-32:GOTO 510
470 IFA=3THENP=2:GOTO 510
480 IFA=5THENP=64:GOTO510
490 IFA=7THENP=-1:GOTO510
500 GOTO700
510 IFVPEEK(MA+P)=0THENPSET(136,8),1:GOT
O590
520 IFVPEEK(MA+P)<>12GOTO 700
530 A=1:IFABS(P)<3THENA=32
540 IFVPEEK(MA+P+A)<>12GOTO 700
550 VPOKEMA,12:VPOKEMA+1,12:VPOKEMA+32,1
2:VPOKEMA+33,12
560 IFP>0THENP=P/2
570 MA=MA+P:VPOKEMA,6:VPOKEMA+1,8:VPOKEM
A+32,7:VPOKEMA+33,9
580 GOTO 700
590 A=P:IFP<0THENA=A*2
600 FORI=0TO6:IFMA+A<>BL!(I)THENNEXT:GOT
O 700
610 IFVPEEK(BL!(I)+P)<>12GOTO700
620 A=1:IFABS(P)<3THENA=32
630 IFVPEEK(BL!(I)+P+A)<>12GOTO 700
640 VPOKEBL!(I),12:VPOKEBL!(I)+1,12:VPOK
EBL!(I)+32,12
650 VPOKEBL!(I)+33,12:VPOKEMA,12:VPOKEMA
+1,12:VPOKEMA+32,12:VPOKEMA+33,12
660 IFP>0THENP=P/2
670 MA=MA+P:BL!(I)=BL!(I)+P
680 VPOKEBL!(I),0:VPOKEBL!(I)+1,0:VPOKEB
L!(I)+32,0:VPOKEBL!(I)+33,0
690 VPOKEMA,6:VPOKEMA+1,8:VPOKEMA+32,7:V
POKEMA+33,9
700 FORI=0TO4:IFFA!(I)=0GOTO 780
710 P=F%(I):IFP>0THENP=P*2
720 IFVPEEK(FA!(I)+P)<>12GOTO 790
730 A=1:IFABS(P)<3THENA=32
740 IFVPEEK(FA!(I)+P+A)<>12GOTO790
```

```
750 VPOKEFA!(I),12:VPOKEFA!(I)+1,12:VPOK
EFA!(I)+32,12:VPOKEFA!(I)+33,12
760 FA!(I)=FA!(I)+F%(I)
770 VPOKEFA!(I),2:VPOKEFA!(I)+1,4:VPOKEF
A!(I)+32,3:VPOKEFA!(I)+33,5
780 NEXT:TI=TI-1:GOSUB1210:IFTI=0GOTO 10
60ELSEGOTO 450
790 IFP>0THENP=P/2ELSEP=P*2
800 P=P*-1
810 IFVPEEK(FA!(I)+P)<>12GOTO 850
820 A=1:IFABS(P)<3THENA=32
830 IFVPEEK(FA!(I)+P+A)<>12GOTO 850
840 F%(I)=F%(I)*-1:GOTO 750
850 IFABS(P)<3GOTO 920
860 IFVPEEK(FA!(I)-1)<>12GOTO890
870 IFVPEEK(FA!(I)+31)<>12GOTO 890
880 F%(I)=-1:GOTO 750
890 IFVPEEK(FA!(I)+2)<>12GOTO 980
900 IFVPEEK(FA!(I)+34)<>12GOTO980
910 F%(I)=1:GOTO 750
920 IF VPEEK(FA!(I)-32)<>12GOTO 950
930 IF VPEEK(FA!(I)-31)<>12GOTO950
940 F%(I)=-32:GOTO 750
950 IF VPEEK(FA!(I)+64)<>12GOTO980
960 IFVPEEK(FA!(I)+65)<>12GOTO 980
970 F%(I)=32:GOTO750
980 SOUND7,247:SOUND8,16:SOUND13,0:SOUND
6,15
990 SOUND 11,100:SOUND12,10:FK=FK-1
1000 VPOKEFA!(I),12:VPOKEFA!(I)+1,12:VPO
KEFA!(I)+32,12:VPOKEFA!(I)+33,12
1010 FA!(I)=0:SC=SC+10:GOSUB1190:IFFK<>0
GOTO0780
1020 SOUND7,56:ST=ST-100:IFST=200THENST=
1000
1030 PLAY"S11M5000T255L803G05G03F05A03E0
5B03D06CL403C06CC"
1040 SC=SC+INT(TI/2):GOSUB1190
1050 IFPLAY(0)GOTO1050ELSE300
1060 SOUND7,56:PLAY"t120s8m518o4g.ga16g.
l16gabo5c8"
```

```
1070 IFPLAY(0)THEN1070
1080 FORI=0TO1000:NEXT
1090 LINE(0,8)-(255,15),4,BF
1100 PSET(88,8),4:COLOR15:PRINT#1,"FIM D
E JOGO"
1110 FORI=0TO2000:NEXT:IFSC>HSTHENHS=SC
1120 LINE(0,8)-(255,15),4,BF
1130 PSET(48,8),4:COLOR10:PRINT#1,"TECLE
 ESPACO PARA COMECAR"
1140 IF STRIG(0)=0GOTO1140:LINE(0,8)-(25
5,15),4,BF
1150 LINE(0,8)-(255,15),4,BF
1160 PSET(80,8),4:COLOR15:A=HS:MS=5:GOSU
B1230:PRINT#1,"RECORDE :";S$
1170 FORI=0TO1000:NEXT:BEEP
1180 LINE(0,8)-(255,15),1,BF:GOTO290
1190 LINE(64,8)-(118,15),1,BF
1200 A=SC:M=5:GOSUB1230:PSET(16,8),1:PRI
NT#1,"SCORE:";S$:RETURN
1210 LINE(184,8)-(215,15),1,BF
1220 A=TI:M=4:GOSUB1230:PSET(136,8),1:PR
INT#1,"TEMPO:";S$:RETURN
1230 S$=STR$(A):S$=RIGHT$("0000"+RIGHT$(
S$,LEN(S$)-1),M):RETURN
1240 FORI=&H1840TO&H1AFF:VPOKEI,12:NEXT:
RETURN
1250 SCREEN2,2,0:COLOR 15,1:CLS
1255 OPEN"grp:"AS#1:LINE(0,0)-(255,7),10
,BF
1260 PSET(80,0),11:COLOR 13:PRINT#1,"BAL
LOON CRASH"
1270 PSET(16,16),1:COLOR 15:PRINT#1,"Man
eje o garoto, mova os blo- cos"
1280 PSET(16,32),1:PRINT#1,"e estoure os
 baleezinhos."
1290 PSET(16,48),1:PRINT#1,"O balao,ao s
er imobilizado..."
1300 PSET(16,64),1:PRINT#1,"...acaba est
ourando."
1310 PSET(16,80),1:PRINT#1,"O garoto so
desloca 1 bloco de cada vez."
```

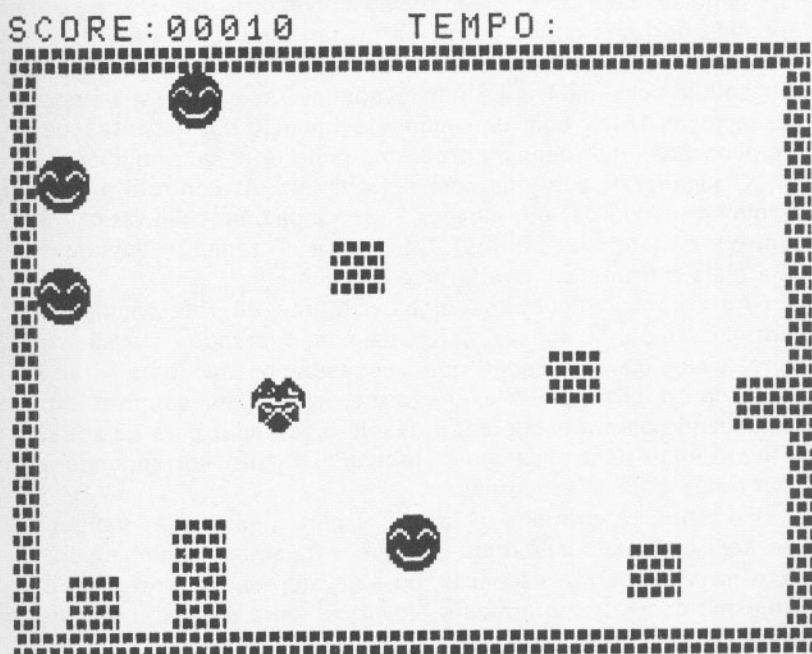
```
1320 PSET(16,96),1:PRINT#1,"E o jogo aca
ba quando o tempo  terminar"
1330 PSET(40,128),1:COLOR 8:PRINT#1,"TEC
LE ESPACO PARA COMECAR"
1340 IFSTRIG(0)=0GOTO 1340ELSECOLOR 15:C
LS:RETURN
```

ANÁLISE

Nas linhas de 100 a 190 estão contidos os dados necessários para a confecção do desenho dos balões, do balomaníaco (você!), dos tijolos e das paredes. Esses dados podem ser alterados a seu critério para produzir outros padrões.

As instruções que aparecem no início do programa (linhas de 1260 a 1330) podem ser alteradas ou eventualmente eliminadas do mesmo, sem maiores consequências.

Nas linhas de 450 a 490 o computador identifica a tecla pressionada e direciona o movimento do balomaníaco (através da função STICK).

O resto do programa deixaremos para que você mesmo analise.

TRON

INSTRUÇÕES

Além de ser o nome de um utilíssimo comando do BASIC-MSX, é também o nome de um dos personagens do filme que, coincidentemente possui o nome TRON.

Quem teve o prazer de ver o filme, certamente deve ter gostado das cenas montadas por computador, em que o herói se digladia com inimigos pilotando motos que deixam atrás de si rastros sólidos.

A mesma emoção sentirá quem tiver o prazer de pilotar este jogo no seu MSX, mesmo não correndo o perigo de ser 'apagado por uma curva imprecisa ou um túnel sem saída!!

Já deve ter dado para perceber que o objetivo do jogo é não colidir com nada, mas não é apenas isso. A real emoção de se jogar TRON com um oponente humano são as situações e emboscadas que nenhum programa pode criar ou simular.

O jogador 1, que joga com o "joystick" A, controla a moto vermelha (esquerda) e o jogador 2, que joga com o "joystick" B, controla a moto azul (direita). Caso você só tenha 1 "joystick", veja mais à frente como alterar o programa.

Ao contrário de outros jogos, o ponto de referência está dentro da moto, como se estivéssemos pilotando. Dessa maneira, os únicos comandos que necessitamos são para virar à esquerda e à direita. Por exemplo: se, na tela, nossa moto estiver andando para a esquerda, e desejarmos andar para baixo, se estivéssemos dentro da moto, este movimento corresponderia a virarmos para a esquerda!

Portanto, se virarmos o "joystick" para a esquerda, qualquer que seja a direção e sentido em que estejamos caminhando, a moto irá virar para a esquerda, ou seja, no sentido anti-horário. O mesmo ocorre se virarmos o "joystick" para a direita: a moto virará no sentido horário.

Figura 5.1 — Programa TRON

TRON

```
10 ' T R O N
20 ' Fernando da Costa Grossi
30 '
10000 OPEN"GRP:" FOR OUTPUT AS#1
10010 KEY OFF
10020 GOSUB 10390
10030 DEFINT A-Z: RESTORE
10040 X1=10: Y1=96
10050 X2=244: Y2=96
10060 V1=2: V2=0
10070 D1=0: D2=0
10080 S1=255: S2=255
10090 FOR X=1 TO 7:READ A,B: SOUNDA,B: N
EXT X
10100 SCREEN3: COLOR 15,1,7: CLS
10110 LINE(0,0)-(255,191),,B
10120 SOUND 7,60
10130 A=STICK(1): SOUND 0,S1:S1=S1-4
10140 B=STICK(2): SOUND 0,S2:S2=S2-4
10150 IF A=7 AND D1=0 THEN S1=255: D1=1:
V1=V1+1
10160 IF B=7 AND D2=0 THEN S2=255: D2=1:
V2=V2+1
10170 IF A=3 AND D1=0 THEN S1=255: D1=1:
V1=V1-1
10180 IF B=3 AND D2=0 THEN S2=255: D2=1:
V2=V2-1
10190 IF A=0 THEN D1=0
10200 IF B=0 THEN D2=0
10210 IF V1<0 THEN V1=3 ELSE IF V1>3 THE
N V1=0
10220 IF V2<0 THEN V2=3 ELSE IF V2>3 THE
N V2=0
10230 IF V1=0 THEN X1=X1-4 ELSE IF V1=1
THEN Y1=Y1+4 ELSE IF V1=2 THEN X1=X1+4 E
LSE IF V1=3 THEN Y1=Y1-4
```

```
10240 IF V2=0 THEN X2=X2-4 ELSE IF V2=1
THEN Y2=Y2+4 ELSE IF V2=2 THEN X2=X2+4 E
LSE IF V2=3 THEN Y2=Y2-4
10250 IF POINT(X2,Y2)<>1 THEN 10300
10260 IF POINT(X1,Y1)<>1 THEN 10310
10270 PSET(X1,Y1),8
10280 PSET(X2,Y2),4
10290 GOTO 10130
10300 A$="Jogador 1 venceu":GOTO 10320
10310 A$="Jogador 2 venceu":GOTO 10320
10320 SOUND 7,7: FOR X=0 TO 31: SOUND6,X
:COLOR XMOD16: NEXT X:SOUND 7,63: COLOR,
,5
10330 FOR X= 0 TO 1000: NEXT X: COLOR 1,
15,15: SCREEN2
10340 PRESET(56,88):PRINT#1,A$
10350 PRESET(57,88):PRINT#1,A$
10360 IF STRIG(0)=0 AND STRIG(1)=1 AND S
TRIG(2)=0 THEN GOTO 10360
10370 GOTO 10030
10380 DATA 7,255,8,15,9,15,1,1,3,1,0,255
,2,255
10390 SCREEN 2: COLOR 0,8,8: CLS: PI=4*A
TN(1)
10400 DRAW"S32 BM64,80 R4DLD3L2U3LU BM96
,96 D2RU2LR2F2LH2 BM112,80 LDRBFR"
10410 CIRCLE(112,96),16,0,0,PI/2: CIRCLE
(112,96),8,0,0,PI/2:CIRCLE(144,96),16,0
10420 DRAW"BM160,80 D4RU2R2U2RD4H4"
10430 PAINT(65,81),0:PAINT(105,81),0: PA
INT(97,97),0:PAINT(106,97),0:PAINT(129,9
7),0:PAINT(161,97),0
10435 PAINT(190,97),0
10440 COLOR1,8,4
10450 PRESET(33,170)
10455 PRINT#1,"Fernando da Costa Grossi"
: PRESET(34,170)
10456 PRINT#1,"Fernando da Costa Grossi"
10460 IF INKEY$<>CHR$(13) THEN 10460 ELS
E RETURN
10470 RETURN
```

ANÁLISE

O programa se divide em três blocos bem distintos:

linha 10000 a 10120: é a parte de inicialização das variáveis do programa. Note que a maioria das variáveis são duplicadas, pois o jogo controla 2 motos.

linha 10130 a 10370: é o laço principal do jogo, onde as motos são controladas, as colisões analisadas e os créditos são dados ao vencedor.

linha 10380 a 10470: são as linhas que montam a apresentação. Todo o desenho é montado com a cor invisível, e somente depois de pronto a cor é mudada.

Observem as linhas 10130 e 10140, com o comando STICK, que estão ajustados para os "joysticks" A e B, respectivamente.

Caso você queira alterar alguns dos jogadores para o teclado, basta alterar o primeiro ou o segundo comando STICK para STICK(0).

JOGOS DE INTELIGÊNCIA

Nos jogos de inteligência, normalmente, o adversário do operador é o próprio computador. Na realidade o duelo de inteligências não é entre o Homem e a Máquina como muitas pessoas, afetadas pelo "complexo de Frankenstein", poderiam crer.

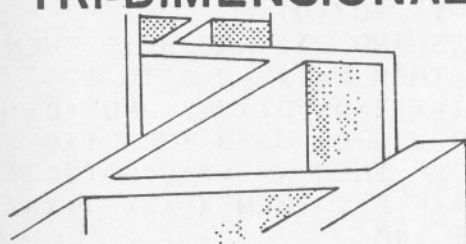
O verdadeiro confronto é entre a mente do jogador e a do programador. Obviamente o programador tem a disposição a memória sem falhas e a incrível rapidez do micro, para suprir a inteligência que falta à máquina.

Nesta seção são discutidos alguns jogos inteligentes, assim denominados mais por exigirem inteligência do jogador do que por simularem inteligência no computador. Apenas o último, OTHELLO, tem esta característica: o computador parece "pensar" as jogadas e, às vezes, nos surpreende com lances inesperados.

NO LABIRINTO e na BATALHA NAVAL espera-se que o leitor desenvolva a chamada INTELIGÊNCIA ESPACIAL, normalmente tão esquecida no ensino tradicional.

No MEMÓRIA é reproduzido o clássico jogo que costuma ser realizado com cartões e com ele se objetiva o desenvolvimento da chamada MEMÓRIA VISUAL.

O FORCABAS, apesar do nome, é um jogo extremamente original. O computador propõe à adivinhação do leitor palavras reservadas do BASIC MSX (incluindo o DISK BASIC) que ele lê em sua própria ROM.

LABIRINTO TRI-DIMENSIONAL

INSTRUÇÕES

Este programa é um simulador de labirintos tridimensionais que reproduz com grande realismo a visão de uma pessoa colocada em seu interior. O objetivo do jogador é caminhar através de suas passagens à procura da saída, de uma forma muito semelhante à de um labirinto real.

A caracteristica mais marcante deste jogo é o efeito de profundidade (ou tridimensionalidade) conseguido pela apresentação perspectivada das paredes e passagens laterais do corredor visualizado. Em qualquer instante, o jogador deve ter em mente a posição na qual está e a direção para a qual ele está voltado (Norte, Sul, Leste ou Oeste).

É interessante notar que na verdade, o labirinto não é tridimensional, mas bidimensional. Mesmo assim, ele é dito tridimensional devido ao efeito de perspectiva causado pelo programa.

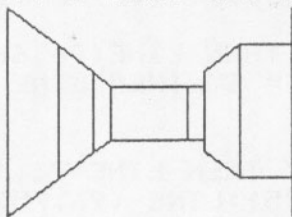
Para jogar, comande RUN, e após alguns segundos aparecerá na tela a parte do labirinto visível da sua posição. Você dispõe então das seguintes opções: avançar (tecla ▲) virar para a esquerda (tecla ◀), virar para a direita (tecla ▶) e mapa (tecla ▼). Na opção mapa é indicada a planta do labirinto e a sua posição, bem como a sua direção. A saída se torna visível quando você está voltado para a mesma, a menos de cinco passos dela.

Figura 6.1 — Programa LABIRINTO

```
10 REM LABIRINTO
20 REM MILTON MALDONADO JR.
30 REM ABRIL DE 1986
40 CLS:CLEAR1000:OPEN "GRP:" FOR OUTPUT
AS 1:DIM A$(19,25),X%(400),Y%(400),DX(3)
,DY(3):FOR L=0 TO 3:READ DX(L),DY(L),D$(
L):NEXT L:P=0:X=2:Y=2:PRINT"Aguarde - pr
ocessando o labirinto"
```

```
50 X%(P)=X:Y%(P)=Y
60 D=INT(4*RND(-TIME))
70 I=X+2*DX(D):J=Y+2*DY(D)
80 IF I>1 AND I<25 AND J>1 AND J<19 THEN
 IF A$(J,I)<>"1" THEN X=I:Y=J:GOTO 120
90 FOR D=0 TO 3:I=X+2*DX(D):J=Y+2*DY(D):
IF I<2 OR I>24 OR J<2 OR J>18 THEN 110
100 IF A$(J,I)<>"1" THEN X=I:Y=J:GOTO120
110 NEXT D:P=P-1:IF P>1 THEN X=X%(P):Y=Y
%(P):GOTO 60 ELSE 140
120 P=P+1:A$(Y,X)="1":A$(Y-DY(D),X-DX(D)
)="1":GOTO 50
130 DATA 1,0,LESTE,0,-1,NORTE,-1,0,OESTE
,0,1,SUL
140 X=2:Y=2:D=3:A$(19,24)="S"
150 GOSUB 450
160 SCREEN 2:CLS:ES=(D+1)MOD4:DI=D-1:IF
DI<0 THEN DI=3
170 FOR I=0 TO 4:E$=A$(Y+I*DY(D)+DY(ES),
X+I*DX(D)+DX(ES)):D$=A$(Y+I*DY(D)+DY(DI)
,X+I*DX(D)+DX(DI)):F$=A$(Y+(1+I)*DY(D),X
+(1+I)*DX(D))
175 ON I+1 GOSUB 250,290,330,370,410
180 IF F$="1" THEN NEXT I
190 C=STICK(0):IF C=0 THEN 190
200 IF C=3 THEN D=D-1:IF D<0 THEN D=3
210 IF C=7 THEN D=D+1:IF D=4 THEN D=0
220 IF C=1 AND A$(Y+DY(D),X+DX(D))="1" T
HEN X=X+DX(D):Y=Y+DY(D):C=0
230 IF C=5 THEN GOSUB 450:SCREEN 2
240 GOTO 160
250 IF E$<>"" THEN LINE(10,153)-(46,153)
:LINE-(46,27):LINE-(10,27) ELSE LINE (10
,180)-(46,153):LINE-(46,27):LINE-(10,0)
260 IF D$<>"" THEN LINE(246,153)-(210,15
3):LINE-(210,27):LINE-(246,27) ELSE LINE
 (246,180)-(210,153):LINE-(210,27):LINE-
(246,0)
270 IF F$="" THEN LINE (46,27)-(210,153)
,,B ELSE IF F$="S" THEN LINE (46,27)-(21
0,153),,BF
```

```
280 RETURN
290 IF E$<>"" THEN LINE(46,130)-(76,130)
:LINE-(76,50):LINE-(46,50) ELSE LINE (46
,153)-(76,130):LINE-(76,50):LINE-(46,27)
300 IF D$<>"" THEN LINE(210,130)-(180,13
0):LINE-(180,50):LINE-(210,50) ELSE LINE
 (210,153)-(180,130):LINE-(180,50):LINE-
(210,27)
310 IF F$="" THEN LINE(76,50)-(180,130),
,B ELSE IF F$="S" THEN LINE (76,50)-(180
,130),,BF
320 RETURN
330 IF E$<>"" THEN LINE(76,114)-(96,114)
:LINE-(96,66):LINE-(76,66) ELSE LINE (76
,130)-(96,114):LINE-(96,66):LINE-(76,50)
340 IF D$<>"" THEN LINE(180,114)-(160,11
4):LINE-(160,66):LINE-(180,66) ELSE LINE
 (180,130)-(160,114):LINE-(160,66):LINE-
(180,50)
350 IF F$="" THEN LINE(96,66)-(160,114),
,B ELSE IF F$="S" THEN LINE (96,66)-(160
,114),,BF
360 RETURN
370 IF E$<>"" THEN LINE(96,107)-(106,107
):LINE-(106,75):LINE-(96,75) ELSE LINE (
96,114)-(106,107):LINE-(106,75):LINE-(96
,66)
380 IF D$<>"" THEN LINE(160,107)-(150,10
7):LINE-(150,75):LINE-(160,75) ELSE LINE
 (160,114)-(150,107):LINE-(150,75):LINE-
(160,66)
390 IF F$="" THEN LINE (106,75)-(150,107
),,B ELSE IF F$="S" THEN LINE (106,75)-(
150,107),,BF
400 RETURN
410 IF E$<>"" THEN LINE(106,103)-(111,10
3):LINE-(111,80):LINE-(106,80) ELSE LINE
 (106,107)-(111,103):LINE-(111,80):LINE-
(106,75)
420 IF D$<>"" THEN LINE(150,103)-(145,10
3):LINE-(145,80):LINE-(150,80) ELSE LINE
 (150,107)-(145,103):LINE-(145,80):LINE-
(150,75)
```

```
430 IF F$="" THEN LINE(111,80)-(145,103)
,,B ELSE IF F$="S" THEN LINE (111,80)-(1
45,103),,BF ELSE PRESET (126,86):PRINT #
1,"?"
440 RETURN
450 SCREEN 0:PRINT "Planta do labirinto"
:PRINT:A$(Y,X)="*":FOR I=1 TO 19:FOR J=1
 TO 25:IF A$(I,J)="1" THEN PRINT " "; EL
SE IF A$(I,J)="*" THEN PRINT"*"; ELSE IF
 A$(I,J)="S" THEN PRINT CHR$(204); ELSE
PRINT CHR$(219);
460 NEXT:PRINT:NEXT:A$(Y,X)="1":PRINTD$(
D)
470 IF STICK(0)=0 THEN 470 ELSE RETURN
```

ANÁLISE

O programa começa criando na memória um espaço bidimensional de 19x25 posições através do "array" AS(19.25). Inicialmente, entretanto, este espaço está "fechado", ou seja, está todo emparedado. O programa então começa a "cavar túneis" na parede sólida até ter aberto passagens suficientes para a realização do jogo.

O algorítmo de abertura de túneis (linhas 50 a 120) é particularmente interessante, pois sempre consegue unir por um caminho contínuo a entrada e a saída, e a espessura das paredes é sempre igual à espessura das passagens. E, quando o labirinto é terminado, o jogador pode decorar o desenho do mapa através de uma planta que é mostrada na tela, pois isto será importante na hora de tentar sair do labirinto.

A movimentação do jogador é feita variando-se suas coordenadas X e Y em uma das direções: leste (D=0), norte (D=1), oeste (D=2) e sul (D=3), através dos vetores DX(D) e DY(D). Para saber se o movimento do jogador é válido (isto é, se não colidiu com uma "parede"), é feito um teste na linha 220 que

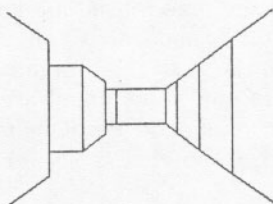
habilita ou não o incremento das coordenadas.

O leitor mais exigente certamente não ficará satisfeito com esta versão "protótipo" do jogo. Uma apresentação inicial se faz conveniente, bem como uma tela de congratulações pela saída do labirinto. E, para os mais curiosos, existe a possibilidade de se visualizar o processo de montagem do labirinto em tempo real.

A segunda listagem trás estas alterações, às quais podem ser acrescentadas outras ao gosto do leitor. Que tal fazer, por exemplo, o fugitivo aparecer inicialmente em uma parte aleatório do labirinto?

Lembrete: use X e Y pares, para não haver o risco do fugitivo "cair" numa parede... Outra sugestão é fazer a direção inicial ser aleatória Norte, Sul, Leste ou Oeste?) e, para complicar mais ainda, bloquear a visualização da planta durante o jogo...

Figura 6.2 — Algumas implementações

```
45 PRINT:FOR I=1 TO 19:PRINT " ";STRING$
(25,219):NEXT
110 NEXT D:P=P-1:IF P>1 THEN X=X%(P):Y=Y
%(P):LOCATE X,Y+1:PRINT"*";CHR$(8);:FORW
=0TO10:NEXTW:PRINT" ":GOTO 60 ELSE 140
120 P=P+1:A$(Y,X)="1":LOCATE X,Y+1:PRINT
" ":A$(Y-DY(D),X-DX(D))="1":LOCATE X-DX(
D),Y-DY(D)+1:PRINT" ":GOTO 50
225 IF C=1 AND A$(Y+DY(D),X+DX(D))="S" T
HEN 1000
226 P=P+1
1000 SCREEN 0:PRINT "        C O N G R A T
U L A C O E S":PRINT:PRINT"Voce consegui
u sair em";P;"passos.":PRINT:PRINT"Joga
novamente ?"
1020 A$=INPUT$(1):IF A$="s" OR A$="S" TH
EN RUN
1030 IF A$="n" OR A$="N" THEN END ELSE 1
020
```

INSTRUÇÕES

Um dos programas mais tradicionais para microcomputadores é o "Jogo da Forca", cujas regras pressupomos conhecidas. A maioria desses programas apenas monitoram o jogo entre duas pessoas: uma introduz a palavra e a outra tenta adivinhar.

Alguns poucos programas conseguem armazenar algumas palavras e depois sortear uma delas para que um único jogador tente adivinhar. Em geral, esses programas são bem simples e fáceis de se fazer. Talvez por isso, um dos primeiros jogos que qualquer programador iniciante tenta fazer assim que adquire um microcomputador, seja esse.

A seguir, apresentamos a nossa versão do "Jogo da Forca" para o MSX.

Ele utiliza vários recursos da máquina, como cor, som, movimentos (com SPRITEs), etc, mas o que mais o diferencia de outras versões é que ele joga sorteando palavras do BASIC MSX armazenadas na ROM. Assim, ao jogar com o micro, o usuário além de se entreter fica conhecendo melhor o vocabulário do BASIC MSX.

Outra diferença é que o réu (a ser enforcado) não é construído aos poucos (primeiro a cabeça, depois um braço, etc) mas aparece desde o início de corpo inteiro. A cada erro cometido, ele se aproxima mais da forca.

Digite o programa com atenção. Ao introduzir as linhas de 418 a 740 redobre o cuidado.

Para jogar, vá introduzindo as letras que você crê que façam parte da palavra incógnita. A cada erro cometido, o réu (você) se aproximará mais da corda. Ao ser enforcado, a palavra incógnita será apresentada para que sua "alma" a veja!

Enquanto estiver a caminho do Paraíso, sua alma escutará uma bela marcha fúnebre. Caso você seja inocentado, uma música festiva será tocada.

Figura 7.1 — Programa FORCABAS

```
10 '- FORCABAS --------------------
20 '- Renato da Silva Oliveira -
30 '---------------------------------
40 ' INTRODUZ A PALAVRA
50 '#################################
60 POKE &HFCAB,1
70 COLOR1,7,7:SCREEN0,,0:CLEAR4000
80 LOCATE4,4:PRINT"F O R C A":PRINT
90 PRINT" DIGITE RETURN E ESPERE ALGUNS
         INSTANTES.":KEYOFF
100 INPUTA$:BEEP:S$="":INTERVALON
110 ONINTERVAL=6GOSUB1460
120 A=65536!*RND(-TIME)
130 A%=163*RND(A)+1
140 EN=14962:A$="":I=65:C=0
150 A$=A$+CHR$(I)
160 P=PEEK(EN):Q=PEEK(EN+1):P$=CHR$(P)
170 IF P<128 THEN A$=A$+P$:GOTO 220
180 A$=A$+CHR$(P-128)
190 EN=EN+1:C=C+1
200 IFC=A%THEN280
210 IFPEEK(EN+1)<>0THENA$="":A$=A$+CHR$(
I)
220 IFPEEK(EN)<>0THEN260
230 A$="":I=I+1:Q$=CHR$(I)
240 IFQ$="J"ORQ$="Q"THEN 260
250 A$=A$+Q$
260 EN=EN+1
270 IF EN<=15649 THEN 160
280 IFTE<200THEN280ELSEINTERVALOFF
290 S$=S$+" ":IFLEN(S$)<LEN(A$)THEN290
300 C$=" "
310 ' FAZ A TELA E A FORCA
320 '#################################
330 COLOR1,7,7:SCREEN 2,2,0:KEY OFF
340 OPEN"GRP:"FOR OUTPUTAS#1
350 FOR F=2 TO LEN(A$)+1
360 LINE(8*F-2,29)-(8*F+6,40),1,B
370 PRESET(8*F,40):PRINT#1,"_"
380 NEXT F
```

```
390 LINE(20,191)-(200,80),9,BF
400 PSET(50,190),11
410 DRAW"e1r1e1r1e1r1u90r35d2"
420 DRAW"l33d88f1r1f1r1f1r1l13"
430 DRAW"bm59,164r55d2l55"
440 DRAW"bm104,190e1r1e1r1e1r1u20r2"
450 DRAW"d20f1r1f1r1f1r1l13"
460 DRAW"bm+4,-4h19r2f17"
470 DRAW"bm-50,+2e19l2g17"
480 DRAW"bm+0,-72e12l2g10"
490 DRAW"bm+25,+0u10d30f2d3g1l2"
500 DRAW"h1u3e2r1u3"
510 DRAW"bm113,168r15d6r15d6r15d6r15"
520 DRAW"d5l1u4l15u6l15u6l15u6l14"
530 DRAW"bm77,190e1r1e1r1e1"
540 DRAW"r3f1r1f1r1f1r1l14"
550 DRAW"bm+7,-4e12f1g11"
560 DRAW"bm-3,+0h12g1f12"
570 FOR F=1 TO 32
580 READ C$:A=VAL("&H"+C$)
590 D$=D$+CHR$(A):NEXT F
600 SPRITE$(1)=D$:D$=""
610 FOR F=1 TO 32
620 READ C$:A=VAL("&H"+C$)
630 D$=D$+CHR$(A):NEXT F
640 SPRITE$(2)=D$:D$=""
650 PUT SPRITE2,(173,160),1,1
660 PUT SPRITE3,(173,175),4,2
670 DATA3,7,5,7,2,1,3f,7F
680 DATA7E,6F,6E,6F,66,37,1A,F
690 DATAC0,E0,A0,E0,40,80,FC,FE
700 DATA76,F6,76,F6,66,EC,58,F0
710 DATA7,7,7,7,7,7,7,6
720 DATA6,3,3,3,3,3,3,1D
730 DATAE0,E0,E0,E0,E0,E0,E0,60
740 DATA60,C0,C0,C0,C0,C0,C0,B8
750 ' TENTATIVAS DO ACUSADO
760 '####################################
770 B=8:IFLEN(A$)>8THENB=LEN(A$)
780 FOR F=1 TO B
790 ' VERIFICA SE ACERTOU
```

```
800 '##################################
810 IF S$=A$ THEN 1380
820 ' ENTRA COM A LETRA
830 '##################################
840 B$=INKEY$:IF B$="" THEN 840
850 BEEP:IF LEN(B$)<>1 THEN 840
860 ' MOSTRA LETRA NO VIDEO
870 '##################################
880 A=0:F$=F$+B$
890 PSET(24,84),9:PRINT#1,F$
900 ' VERIFICA LETRA CERTA
910 '##################################
920 IP=0:FOR G=1 TO LEN(A$)
930 IF MID$(A$,G,1)<>B$ THEN 960
940 PRESET(8*(1+G),32):PRINT#1,B$
950 MID$(S$,G,1)=B$:A=A+1:IP=1
960 NEXT G:F=F-IP
970 IF A<>0 THEN 1350
980 ' MOVE O REU
990 '##################################
1000 IFF=1THENPUTSPRITE2,(159,153),1,1:P
UTSPRITE3,(159,169),4,2
1010 IFF=2THENPUTSPRITE2,(143,147),1,1:P
UTSPRITE3,(143,163),4,2
1020 IFF=3THENPUTSPRITE2,(127,141),1,1:P
UTSPRITE3,(127,157),4,2
1030 IFF=4THENPUTSPRITE2,(111,135),1,1:P
UTSPRITE3,(111,151),4,2
1040 IFF=5THENPUTSPRITE2,(95,131),1,1:PU
TSPRITE3,(95,147),4,2
1050 IFF=6THENPUTSPRITE2,(85,131),1,1:PU
TSPRITE3,(85,147),4,2
1060 IFF=7THENPUTSPRITE2,(76,131),1,1:PU
TSPRITE3,(76,147),4,2
1070 IF F<>8 THEN1350
1080 ' MORTE DO REU
1090 '##################################
1100 PLAY"t110m100s0v15","t110v15"
1110 PLAY"o1bbbbo2dc#c#o1bba#br8","o3bbb
bo4dc#c#o3bba#br8"
1120 PLAY"o1bbbbo2dc#c#o1bba#br8","o3bbb
bo4dc#c#o3bba#br8"
```

```
1130 PLAY"o2ddddfeeddc#dr8","o4ddddfeedd
c#dr8" :  PLAY"bagf#f#d","bagf#f#d"
1140 PLAY"bagf#f#d","bagf#f#d"
1150 PLAY"o1bbbbo2dc#c#o1bba#br16","o3bb
bbo4dc#c#o3bba#br16"
1160 FORF=1TO32
1170 READC$:A=VAL("&H"+C$):D$=D$+CHR$(A)
1180 NEXTF:SPRITE$(3)=D$:D$=""
1190 LINE(78,120)-(88,140),9,BF
1200 COLOR11:DRAW"BM84,120D27"
1210 PUT SPRITE 2,(77,147),1,1
1220 FORF=1TO40STEP2
1230 PUT SPRITE3,(77,165),4,3
1240 FORG=1TO30+5*F:NEXTG
1250 PUT SPRITE3,(77,166),4,2
1260 FORG=1TO90+2*F:NEXTG,F
1270 PSET(83,150),1:PSET(86,150),1
1280 DATA 0,7,F,1F,3F,7E,FC,F0
1290 DATA F0,78,3C,1C,E,7,6,C
1300 DATA 0,E0,E0,F0,F0,F0,70,70
1310 DATA 70,70,70,70,70,70,70,2C
1320 PRESET(20,8):COLOR6
1330 PRINT#1,"ASSIM TERMINAM OS CULPADOS
!":PRESET(16,32):COLOR1:PRINT#1,A$
1340 GOTO 1440
1350 NEXT F
1360 ' FINAL
1370 '###############################
1380 PRESET(10,10)
1390 PLAY"v15t200","v15t200","v15t200"
1400 PLAY"o5co4a#gg#gf","o6co5a#gg#gf"
1410 PLAY"o5cdfedco4aa#ag","o6cdfedco5aa
#ag"
1420 PLAY"gagfdgafaa#o5co4af","gagfdgafa
a#o6co5af"
1430 PRINT#1,"VOCE ESTA LIVRE"
1440 PRESET(14,18)
1450 PRINT#1,"DIGITE RETURN PARA NOVO JO
GO" : LINEINPUTA$:RUN
1460 TE=TE+1:PLAY"O6A#64","O7C#32":RETUR
N
```

ANÁLISE

Os principais blocos do programa são iniciados por linhas REM. Algumas instruções mais relevantes são:

1 — A linha 60 trava a tecla CAPS LOCK em maiúsculas.

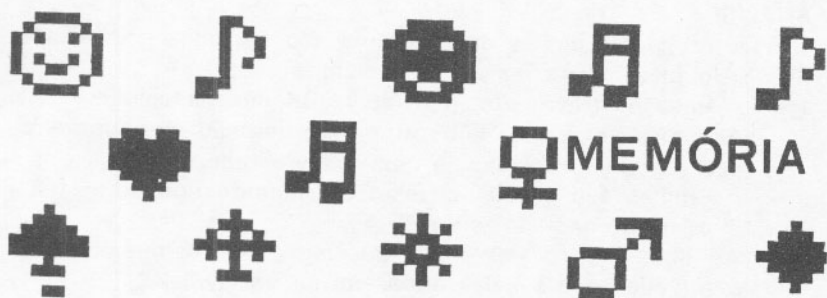
2 — A linha 1490 é uma sub-rotina para geração de sons com eco durante o sorteio de uma palavra reservada.

3 — As linhas 120 e 130 sorteiam um número entre 1 e 163 e o armazenam na variável A%.

4 — As linhas entre 140 e 270 buscam a A%-ésima palavra reservada na ROM e a armazena **na variável A$**.

Os erros mais frequentes ocorrem nas instruções DRAW (desenho da forca), PLAY (músicas para morte e libertação do réu) e DATA (desenho dos SPRITE's).

Caso você tenha algum problema na execução do programa, releia a NOTA DO EDITOR.

A palavra formada pelas letras certas do acusado é armazenada em S$. Quando S$ e A$ forem iguais, o réu é inocentado (linha 810).

MEMÓRIA

INSTRUÇÕES

Este programa é uma adaptação do Jogo da Memória. As principais regras desse jogo estão enumeradas a seguir:

(1) Existem 88 pares de cartões;
(2) Cada cartão tem um caractere desenhado numa das faces e a outra face mascarada de modo a ser indistingüível (como as cartas de um baralho).
(3) Podem participar de 1 a 6 jogadores;
(4) Os cartões são embaralhados com as faces desenhadas para baixo e, depois, são dispostos em 11 linhas de 16 colunas cada;
(5) O jovo inicia com um lance do jogador 1 e depois dele jogam, sucessivamente os jogadores 2, 3, 4, 5 e 6;
(6) Cada lance consiste em desvirar dois cartões. Se eles forem iguais, permanecerão com os desenhos para cima, 1 ponto será acrescentado ao jogador que executou o lance e ele terá direito a outro lance. Se eles forem diferentes, serão desvirados e o próximo jogador deverá fazer seu lance;
(7) O jogo termina quando os 176 cartões (88 pares) estiverem com os desenhos para cima e vencerá o jogador com mais pontos;
(8) Para desvirar um cartão, o jogador deve mover o quadrado vermelho da parte inferior da tela com as teclas de setas e colocá-lo sobre o cartão escolhido. Pressionando a barra de espaços, o cartão sobre o quadrado **será desvirado.**

Ao digitar o programa, cuidado com as multi-instruções! Lembre-se, também, que o apóstrofo usado nas linhas 10, 20, 30, 40, 460, 610 e 770 substitui a instrução REM.

Figura 8.1 — Programa MEMORIA

```
10 '*  M E M O R I A  (20/05/86)  *
20 '*   Renato da Silva Oliveira   *
30 '********************************
40 '#### inicializacao #############
50 COLOR 15,4:SCREEN1,,0:KEYOFF
60 TIME=65536!*RND(-TIME):LOCATE8,10
70 PRINT"M E M O R I A":LOCATE3,15
80 INPUT"Quantos jogadores (1 a 6)";N
90 IF N<1 OR N>6 OR N<>INT(N) THEN 70
100 COLOR15,4,7:SCREEN2,1:DIM M(15,10)
110 OPEN"GRP:" FOR OUTPUT AS #1
120 FOR F=0 TO 255 STEP 16
130 LINE(F,0)-(F,175),7:NEXT F
140 FOR F=0 TO 176 STEP 16
150 LINE(0,F)-(255,F),7:NEXT F
160 LINE(0,176)-(255,191),1,BF
170 PSET(30,180),1:PRINT#1,"AGUARDE!
(embaralhando)"
180 PSET(31,180),1:PRINT#1,"AGUARDE!
(embaralhando)"
190 CT=1:FOR F=0 TO 10
200 FOR G=0 TO 7
210 M(G,F)=CT:M(G+8,F)=CT:CT=CT+1
220 PLAY"l16s0m5000n"+STR$(CT):NEXTG,F
230 FOR F=1 TO 528
240 A=RND(F)*16:B=RND(2*F)*11
250 C=RND(3*F)*16:D=RND(4*F)*11
260 SWAPM(A,B),M(C,D):PLAY"l16s13m300"
270 PLAY"N"+STR$(M(A,B)):NEXT F
280 SPRITE$(1)=STRING$(8,255)
290 J=1:DIM P(N),V(1),W(1,1),X(1,1)
300 LINE(0,176)-(255,191),1,BF
310 PSET(10,180),1
320 PRINT#1," JOGADOR";J;"          PONTOS
";P(J)
330 ERASE V,W,X:GOSUB 460:STRIG(0)OFF
340 PUT SPRITE 1,(X,Y),0,1
350 IF V(0)<>V(1) THEN 400
360 P(J)=P(J)+1:H=H+1:IF H=88 THEN 770
```

```
370 M(X(0,0),X(0,1))=255
380 M(X(1,0),X(1,1))=255
390 GOTO 440
400 FOR G=1 TO 500:NEXT G
410 LINE(W(0,0)+1,W(0,1)+2)-(W(0,0)+15,W
(0,1)+16),4,BF
420 LINE(W(1,0)+1,W(1,1)+2)-(W(1,0)+15,W
(1,1)+16),4,BF
430 J=J+1:IF J>N THEN J=1
440 LINE(70,180)-(250,188),1,BF
450 GOTO 310
460 '#### movimento do cursor #######
470 STRIG(0)ON:ON STRIG GOSUB 610
480 X=112:Y=175:IQ=0:DIM V(1),W(1,1),X(1
,1)
490 PUT SPRITE 1,(X,Y),9,1
500 A=STICK(0):IF A=0 THEN 590
510 X=X-16*(A>1 AND A<5)
520 X=X+16*(A>5)
530 Y=Y-16*(A>3 AND A<7)
540 Y=Y+16*(A<3 OR A=8)
550 IF X<0 THEN X=0
560 IF X>240 THEN X=240
570 IF Y>175 THEN Y=175
580 IF Y<0 THEN Y=-1
590 FOR G=1 TO 20:NEXT G
600 IF IQ<2 THEN 490 ELSE RETURN
610 '#### escolha da peca ##########
620 STRIG(0) OFF
630 C%=X/16:L%=(Y+1)/16:IFL%=11THENL%=10
640 IF M(C%,L%)=255 THEN 760
650 IFIQ=1ANDW(0,0)=XANDW(0,1)=YTHEN760
660 IF Y=175 THEN 760
670 LINE(X+1,Y+2)-(X+15,Y+16),15,BF
680 W(IQ,0)=X:W(IQ,1)=Y
690 X(IQ,0)=C%:X(IQ,1)=L%
700 V(IQ)=M(C%,L%):IQ=IQ+1
710 M$=CHR$(88+M(C%,L%))
720 COLOR 7:PSET(X+4,Y+5),15
730 IFM(C%,L%)>33THENPRINT#1,M$:GOTO750
740 PRINT#1,CHR$(1)+CHR$(64+M(C%,L%))
```

```
750 BEEP:COLOR 15
760 STRIG(0) ON:RETURN
770 '#### rotina de finalizacao ###
780 LINE(0,176)-(255,191),1,BF
790 PSET(30,180),1:PRINT#1,"O JOGO TERMI
NOU !"
800 FOR F=1 TO 1000 : NEXT F
810 SCREEN 1
820 FOR F=1 TO N
830 PRINT "Jogador";F;":";P(F)
840 IF P(F)>MX THEN MX=P(F):J=F
850 NEXT F
860 PRINT:PRINT:PRINT
870 PRINT "O vencedor foi o jogador";J;"
!"
880 PRINT:PRINT:PRINT
890 PRINT"Digite a barra de espacos"
900 PRINT"para novo jogo !!!"
910 IF STRIG(0)=0 THEN 910 ELSE RUN
```

ANÁLISE

As linhas REM dividem o programa em suas partes principais.

Os cartões são armazenados na matriz M(15,10) e os desenhos são simplesmente caracteres do próprio micro (letras, símbolos gráficos, etc).

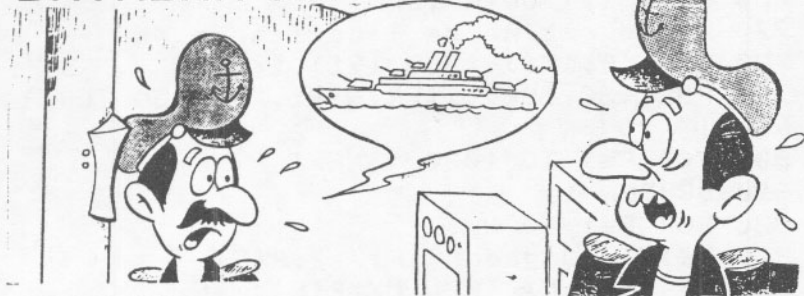
As linhas 170 e 180 fazem um tipo mais cheio de impressão na SCREEN 2.

Após estudar detalhadamente o programa, experimente fazer as alterações sugeridas a seguir:

* Dimensione uma matriz DS(88) e insira em cada um de seus elementos uma sequência de sub-comandos gráficos para serem usados por DRAW. Ao invés de desenhar caracteres (de 1 a 88) nos cartões, desenhe os elementos (de 1 a 88) dessa matriz.

* Faça um par de peças "CORINGAS", que a cada rodada, troquem de lugar com duas outras peças. Isso tornará o jogo mais difícil!

* Tente fazer o jogo na SCREEN 1, redefinindo os caracteres de 1 a 88.

BATALHA NAVAL : 3D

INSTRUÇÕES

Esse jogo não é o tradicional Batalha Naval, já bastante conhecido, onde dois competidores posicionam suas esquadras em um campo quadriculado e cada um tenta afundar os navios do outro.

A finalidade desse jogo é também afundar navios porém, apenas aqueles posicionados pelo computador. Trata-se de um jogo de estratégia, memória e percepção visual.

O computador escolhe aleatoriamente cinco posições e nelas coloca os seus navios. Sua missão é tentar afundá-los. Para isso, eles serão mostrados assim que a tela estiver pronta e você pressionar uma tecla qualquer. Os navios serão, no entanto, mostrados em perspectiva e durante alguns segundos apenas. Você deve, neste intervalo de tempo, avaliar as coordenadas dos navios e memorizá-las pois eles sumirão do seu campo visual. Quando isso acontecer, você terá dez tiros para tentar afundá-los.

Os tiros devem ser dados informando ao computador as coordenadas da posição que você pretende atingir. Informe primeiro a coordenada horizontal (uma letra de A a J) e depois a vertical (um número de 0 a 9) e finalmente confirme o tiro pressionando a tecla "S".

Se o computador não aceitar as coordenadas que você entrou, pressione a tecla "CAPS LOCK" uma única vez e tente de novo.

Caso você erre o tiro será pintada de azul a posição correspondente no mapa quadriculado. Caso acerte, ela será pintada de vermelho e serão computados os pontos. O número de pontos que você ganha por afundar um navio é tanto maior quanto mais navios já tiver afundado e tanto maior quanto mais avançada for a fase em que você se encontra.

Para passar para uma nova fase, todos os cinco navios devem

ser atingidos. Nesse caso, outras posições serão sorteadas para cinco novos navios e o jogo recomeça com a sua pontuação mantida e mais dez tiros.

Quanto mais avançada for a fase, maior será a dificuldade pois menor será o tempo em que os navios serão mostrados.

Caso você dê os dez tiros, numa fase qualquer, e não consiga afundar todos os navios, o jogo termina.

Figura 9.1 — Programa BATALHA NAVAL - 3D

```
100 ' BATALHA NAVAL
110 ' LUIZ TARCISIO DE CARVALHO JR
120 OPEN"grp:"FOROUTPUTAS#1
130 DIMM(10,10)
140 SCREEN2:HI=0:SC=0:Q=0:W=0
150 TA=10: COLOR15,1,1:CLS
160 PSET(206,100),1:COLOR8:PRINT#1,"RECO
RD"
170 PSET(198,110),1:COLOR15:PRINT#1,HI
180 PSET(206,130),1:COLOR8:PRINT#1,"PONT
OS"
190 PSET(198,140),1:COLOR15:PRINT#1,SC
200 PSET(206,160),1:COLOR8:PRINT#1,"FASE
"
210 PSET(198,170),1:COLOR15:PRINT#1,Q+1
220 FORX=152TO232STEP8
230 LINE(X,0)-(X,80),11:NEXTX
240 FORY=0TO80STEP8
250 LINE(152,Y)-(232,Y),11:NEXTY
260 COLOR4: FORX=152TO224STEP8
270 PSET(X+2,85),1:PRINT#1,CHR$((X-152)/
8+65):NEXTX
280 FORY=0TO72STEP8
290 PSET(233,Y),1:PRINT#1,9-(Y/8):NEXTY
300 LINE(50,90)-(150,90),5
310 LINE-(200,190),5
320 LINE-(0,190),5
330 LINE-(50,90),5
340 PAINT(100,189),5
350 COLOR15,1
```

```
360 SPRITE$(1)=CHR$(&H40)+CHR$(&H6)+CHR$
(&H30)+CHR$(&H12)+CHR$(&H58)+CHR$(&H59)+
CHR$(&HFB)+CHR$(&H7B)
370 SPRITE$(2)=CHR$(&HC)+CHR$(&H1E)+CHR$
(&HFF)+CHR$(&H7F)+CHR$(&H0)+CHR$(&H0)+CH
R$(&H0)+CHR$(&H0)
380 GOSUB840
390 FORI=0TO4:X=5+5*Y(I)+X(I)*(19-Y(I)):
Y=180-Y(I)*10:PUTSPRITEI,(X,Y),15,2:NEXT
400 TIME=0:TI=Q*300:IFTI>1200THENTI=1000
410 IFTIME<(1200-TI)THENGOTO410
420 FORI=0TO4:PUTSPRITEI,(0,0),0,2:NEXT
430 GOSUB500:GOSUB450:GOSUB750
440 IFTA=0THENGOTO930ELSEGOTO430
450 TA=TA-1:GOSUB690
460 SOUND6,31:SOUND7,199:SOUND8,16:SOUND
9,16:SOUND10,16:SOUND12,20:SOUND13,0
470 X=5+5*KY+KX*(19-KY):Y=180-KY*10:PUTS
PRITE0,(X,Y),15,1:FORI=1TO200:NEXT
480 PUTSPRITE0,(X,Y),0,1
490 RETURN
500 LINE(30,12)-(140,41),1,BF
510 PSET(30,12),1:PRINT#1,"X ?(A-J) ";
520 GOSUB670
530 K=ASC(A$):IFK>64ANDK<75THEN550
540 GOTO520
550 KX=K-65:PRINT#1,A$
560 PSET(30,20),1:PRINT#1,"Y ?(0-9)";
570 GOSUB670
580 K=ASC(A$):IFK>47ANDK<58THEN600
590 GOTO570
600 KY=K-48:PRINT#1,KY
610 PSET(30,33),1:PRINT#1,"ATIRA ?(S/N)"
;
620 GOSUB670
630 PRINT#1,A$:IFA$<>"S"THEN650
640 RETURN
650 LINE(30,12)-(140,41),1,BF
660 GOTO500
670 A$=INKEY$:IFA$=""THEN670
680 RETURN
```

```
690 LINE(20,55)-(130,63),1,BF
700 IFTA=0THENRETURN
710 PSET(30,55),1
720 FORI=1TOTA
730 PRINT#1,CHR$(1)+CHR$(79);:NEXTI
740 RETURN
750 X1=KX*8+152:Y1=72-KY*8
760 LINE(X1,Y1)-(X1+7,Y1+7),7,BF
770 IFM(KX,KY)=1THEN790
780 COLOR15,1:RETURN
790 M(KX,KY)=0:W=W+1:SC=SC+10*(W+Q)
800 LINE(190,140)-(255,150),1,BF
810 LINE(X1,Y1)-(X1+7,Y1+7),8,BF
820 PSET(198,140),1:PRINT#1,SC
830 COLOR15,1:GOTO780
840 PSET(10,12),1:PRINT#1,"APERTE UMA TE
CLA"
850 IFINKEY$=""THEN850
860 LINE(10,12)-(140,20),1,BF
870 Z=INT(RND(-TIME)*60000!)+1
880 FORI=0TO4:X(I)=INT(RND(Z)*10):Y(I)=I
NT(RND(Z)*10):M(X(I),Y(I))=1:NEXT
890 FORI=0TO3
900 FORJ=I+1TO4
910 IFX(I)=X(J)ANDY(I)=Y(J)THEN870
920 NEXTJ,I:RETURN
930 IFW=5THENW=0:Q=Q+1:GOTO150
940 LINE(50,100)-(150,150),1,BF
950 PSET(60,110),1:PRINT#1,"NOVO JOGO?"
960 PSET(70,130),1:PRINT#1,"S ou N"
970 GOSUB670
980 IFSC>HITHENHI=SC
990 IFA$="S"THENERASEM:SC=0:W=0:Q=0:GOTO
150
```

ANÁLISE

O programa é bem simples quanto à sua lógica e seu detalhamento é o seguinte:

PROGRAMA PRINCIPAL

Linha 120: Abre um arquivo na tela de alta resolução para que possamos escrever textos nela.

Linha 130: dimensiona a matriz M(10,10) que terá cada elemento igual a 1, se existir um navio na posição correspondente a ele, e igual a 0, caso contrário.

Linha 140: seleciona a tela de alta resolução e "zera" as variáveis HI (record), SC (pontos), Q (fase) e W (número de navios afundados).

Linha 150: atribui o valor 10 à variável TA (número de tiros).

Linha 160 a 340: constroem a tela.

Linha 360: define o "sprite" correspondente ao desenho da explosão.

Linha 370: define o "sprite" correspondente ao desenho do navio.

Linha 380: "chama" a sub-rotina 840/920.

Linha 390: calcula as posições dos navios no plano em perspectiva e coloca os "sprites" dos navios.

Linhas 400 e 410: estabelece uma espera tanto menor quanto maior for o número da fase (Q).

Linha 420: "apaga" os "sprites' dos navios.

Linha 430: "chama" as sub-rotinas 500/680, 450/490 e 750/830.

Linha 440: se acabarem os tiros desvia para a linha 930, caso contrário volta a executar a linha 430 ("chamada" das sub-rotinas).

Linha 930 a 990: testam se os cinco navios foram afundados. Caso afirmativo, "zera" W (número de navios afundados), incrementa Q (fase) e desvia a execução do programa para a linha 150. Caso contrário, questiona-se a respeito do desejo de um novo jogo.

SUB-ROTINAS

450/490: dá o tiro, "chama" a sub-rotina 690/740, diminui a variável TA em uma unidade, produz o som do tiro, coloca e retira o "sprite" da explosão nas coordenadas do plano em perspectiva.

500/680: permite a entrada das coordenadas de cada tiro.

690/740: imprime os desenhos correspondentes aos tiros restantes.

750/830: pinta no mapa quadriculado a posição correspondente ao tiro dado. Se houver um navio nessa posição (M(KX,KY) =1), pinta-a de vermelho incrementa e imprime os pontos. Note que o incremento é tanto maior quanto maiores forem a fase e o número de navios já afundados.

840/920: espera a pressão de uma tecla para tornar aleatório o valor da variável TIME. Sorteia as cinco posições correspondentes aos navios e atribui o valor 1 aos respectivos elementos da matriz M(10,10). Verifica também se não houve repetição nas posições sorteadas.

OTHELLO

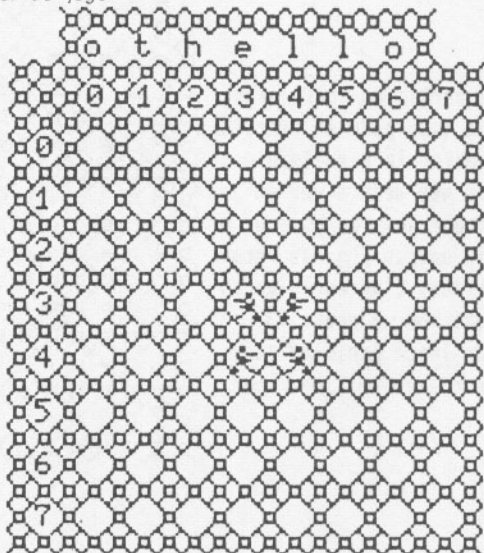
INSTRUÇÕES

Possivelmente o nome deste jogo deve-se a semelhança da estratégia usada pelos competidores e a "estratégia" usada por OTHELLO para matar DESDÊMONA na história de SHAKESPEARE: o estrangulamento.

O jogo transcorre sobre um tabuleiro quadrado (campo de batalha) com 64 casas, onde dois exércitos combaterão pelo domínio das posições até que todas tenham sido ocupadas.

O microcomputador comandará os exércitos brancos e você comandará os exércitos pretos.

Inicialmente, existem apenas dois exércitos de cada cor no tabuleiro (figura 10.1).

Figura 10.1 — Situação inicial do jogo

O objetivo do jogo é conquistar o maior número de posições possível. Quando as 64 posições do tabuleiro estiverem preenchidas, será o vencedor o jogador que tiver mais posições.

Cada jogador só pode ocupar uma posição se obedecer à seguinte regra:

"PARTINDO DE UMA POSIÇÃO JÁ DOMINADA (BASE), CAMINHAR (pelas linhas, pelas colunas ou pelas diagonais) POR SOBRE POSIÇÕES INIMIGAS E OCUPAR UMA POSIÇÃO LIVRE (OBJETIVO)."

Desse modo, ao realizar uma ocupação, cada jogador estará "estrangulando" alguns exércitos inimigos, cercando-os pelas laterais. Os exércitos inimigos estrangulados serão destruídos e substituídos por exércitos aliados. Portanto, ao ocupar uma posição livre, certamente mais de uma posição será dominada.

Veja na figura 10.2 alguns lances permitidos e outros proibidos.

Figura 10.2 — Lances de OTHELLO

Humano:

Micro :

A partir da configuraçãc ao lado pode-se executar muitos lances. Imaginando que seja a vez do jogador humano, observe as 13 únicas jogadas permitidas. Elas estão marcadas por um ponto (•). Um lance proibido seria, por exemplo, jogar em:

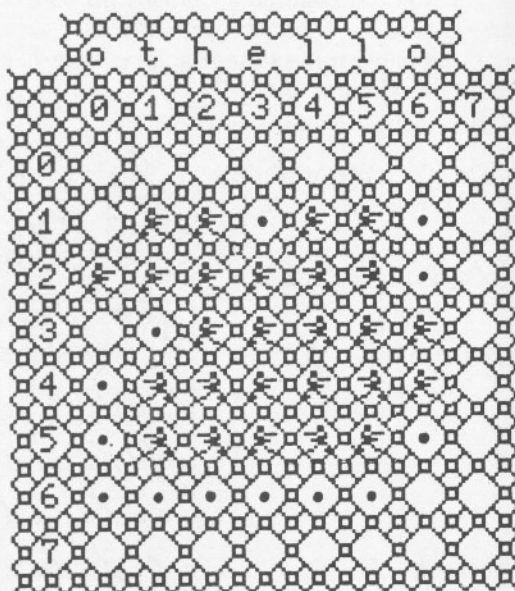
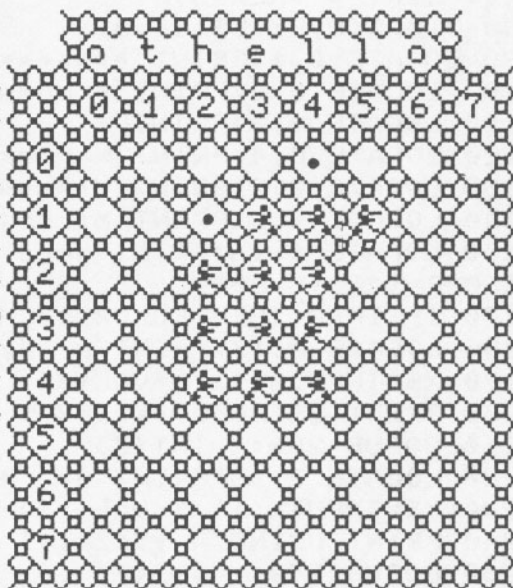
<0,1>

O primeiro movimento é sempre contra o micro e para realizá-lo você deve introduzir as coordenadas x (coluna) e y (linha) da posição livre a ser ocupada. Se para chegar até ela existir mais de uma posição BASE, todos os exércitos inimigos entre cada uma delas e o OBJETIVO serão destruídos e substituídos por exércitos aliados (figura 10.3).

Figura 10.3 — Lance com BASES múltiplas

Podem surgir situações em que não existe nenhuma posição livre em condições de ser ocupada. Nessa situação, você estará sem OBJETIVO e para informar isso ao micro, introduza 8 como valor de x.

Não entraremos em detalhes quanto ao algorítmo usado pelo micro para fazer seus lances. Certamente, ele não é perfeito. Longe disso, faz apenas com que o micro jogue como um jogador medíocre. Apesar disso, ele é suficientemente bom para lhe impingir algumas derrotas enquanto você aprende a jogar!

Ao digitar o programa, tome cuidado para não alterar nenhum dado das linhas DATA. Evite, também, trocar o O por Ø ou vice-versa nas instruções PLAY.

Figura 10.4 — Programa OTHELLO

```
10 '#        O T H E L L O          #
20 '# Renato da Silva Oliveira      #
30 '################################
40 COLOR 4,4,4:SCREEN 1:WIDTH 32
50 KEY OFF:CLEAR 10000
60 FORF=2TO20:FORG=13TO31:LOCATEG,F
70 PRINT"*":NEXTG,F
80 DIMA$(8,8),C$(16,8)
90 LOCATE 0,1:PRINT"************"
100 LOCATE 0,2:PRINT"* Humano:   *"
110 LOCATE 0,3:PRINT"*           *"
120 LOCATE 0,4:PRINT"* Micro :   *"
130 LOCATE 0,5:PRINT"************"
140 LOCATE 1,8:PRINT"Micro: X"
150 LOCATE 1,16:PRINT"Voce ? O"
160 P=1:GOTO 800
170 IF IP=0 THEN COLOR 15,4:GOSUB 1140
180 LOCATE 2,17:PRINT"x= ";CHR$(127);
190 X$=INPUT$(1):X=VAL(X$)
200 PRINT USING"#";X:Z=X+1
210 GOSUB 290:X=INT(Z)
220 LOCATE 2,18:PRINT"y= ";CHR$(127);
230 Y$=INPUT$(1):Y=VAL(Y$)
240 PRINT USING"#";Y:Z=Y+1
250 GOSUB 290:Y=INT(Z)
260 IF A$ (X,Y)=" "THEN 340
270 LOCATE 2,21:PRINT"Novamente"
280 GOTO 170
290 IF Z=9 THEN Z=0:LOCATE 2,21:PRINT"Pa
sse   ":FOR F=1 TO 200: NEXT F
300 LOCATE 2,21:PRINT"          "
310 IF Z<1 THEN 370
320 IF Z<=8 THEN RETURN
330 GOTO 270
340 B$="X":G$="O":P=0:Q=0
350 LOCATE 6,16:PRINT" <";:PRINT USING"#
";X-1;:PRINT":";:PRINT USING"#";Y-1;:PRI
NT">":GOSUB 460
360 LOCATE 7,8:PRINT"?       ":GOSUB 670
```

```
370 B$="O":G$="X":P=1:Q=0
380 FOR S=1 TO 16
390 FOR R=8 TO 1 STEP -2
400 X=VAL(C$(S,R)):IF X=9 THEN 170
410 T=R-1:Y=VAL(C$(S,T))
420 IF A$(X,Y)=" " THEN GOSUB 460
430 IF Q=1 THEN LOCATE 6,8:PRINT" <";:PR
INTUSING"#";X-1;:PRINT":";:PRINT USING"#
";Y-1;:PRINT">":LOCATE 5,16:PRINT"?
 ":GOSUB 1090
440 IF Q=1 THEN 670
450 NEXT R,S
460 FORI=-1TO1
470 FORJ=-1TO1
480 E=X:F=Y:M=X:N=Y
490 IF I=0 THEN IF J=0 THEN 510
500 IF E+I>=1 THEN IF E+I<=8 THEN IF F+J
>=1 THEN IF F+J<=8 THEN 540
510 NEXT J,I
520 IF P=1 OR Q=1 THEN RETURN
530 GOTO 270
540 IF A$(E+I,F+J)<>B$ THEN 510
550 E=E+I:F=F+J
560 IF E>=1 THEN IF E<=8 THEN IF F>=1 TH
EN IF F<=8 THEN 580
570 GOTO 510
580 D$=A$(E,F)
590 IF D$<>B$ THEN 610
600 GOTO 550
610 IF D$=" " THEN 510
620 A$(X,Y)=G$:M=M+I:N=N+J:T$=A$(M,N)
630 IF A$(M,N)=B$ THEN A$(M,N)=G$
640 IF T$=B$ THEN 620
650 Q=1:GOTO 510
660 LOCATE 16,3:PRINT"0*1*2*3*4*5*6*7"
670 Y=0:C=O:FOR J=1 TO 8
680 LOCATE 14,3+J*2
690 PRINT USING "#";J-1;
700 FOR I=1 TO 8
710 PRINT "*";A$(I,J);
720 IF A$(I,J)="O" THEN Y=Y+1
```

```
730 IF A$(I,J)="X" THEN C=C+1
740 NEXT I,J
750 LOCATE 9,2 : PRINTUSING"##";Y
760 LOCATE 9,4 : PRINTUSING"##";C
770 IF Y=0 OR C=0 OR Y+C=64 THEN 900
780 IF P=0 THEN FOR FF=1 TO 100: NEXT FF
 : RETURN
790 GOTO 170
800 FOR J=1 TO 8:FOR I=1 TO 8
810 A$(I,J)=" ":NEXT I,J
820 A$(4,4)="X":A$(5,5)="X"
830 A$(4,5)="O":A$(5,4)="O"
840 RESTORE 870:FOR M=1 TO 16
850 FOR N=1 TO 8:READ SS
860 C$(M,N)=STR$(SS):NEXTN,M
870 DATA 8,8,1,1,8,1,1,8,6,6,3,6,6,3,3,3
,3,4,6,5,5,6,4,3,4,6,3,5,5,3,6,4,3,1,4,1
,5,1,6,1,1,3,1,4,1,5,1,6,8,3,8,4,8,5,8,6
,3,8,4,8,5,8,6,8,3,2,4,2,5,2,6,2
880 DATA 3,7,4,7,5,7,6,7,2,3,2,4,2,5,2,6
,7,3,7,4,7,5,7,6,2,2,7,7,7,2,2,7,1,2,2,1
,7,1,8,2,1,7,2,8,7,8,8,7,9,9,9,9,9,9,9,9
890 GOTO 660
900 IF C>2*Y THEN 1040
910 IF C>1.5*Y THEN 1000
920 IF C>Y THEN 960
930 LOCATE 0,21
940 PRINT "  Fui derrotado!!!"
950 GOTO 1070
960 LOCATE 0,21
970 PRINT" Consegui!!!"
980 PRINT" Eu o venci!!!"
990 GOTO 1070
1000 LOCATE 0,21
1010 PRINT"  Voce perdeu!!!"
1020 PRINT"  Treine mais um pouco!!!"
1030 GOTO 1070
1040 LOCATE 0,21
1050 PRINT "  Eu o venci!!!"
1060 PRINT "  Voce e' um pato!!!"
1070 PRINT"Para novo jogo, digite SPACE.
"
```

```
1080 IF STRIG(0)=0 THEN 1080 ELSE RUN
1090 PLAY"S0M6000T180"
1100 PLAY"L8O4A#R4G#A#G#F#G#R4"
1110 PLAY"G#A#O5C#4O4A#4"
1120 PLAY"G#A#G#F#D#R2"
1130 RETURN
1140 ' redefine caracteres
1150 LOCATE15,0:PRINT"***************"
1160 LOCATE15,1:PRINT"*o t h e l l o*"
1170 LOCATE15,2:PRINT"***************"
1180 DATA 00110000
1190 DATA 00110000
1200 DATA 01001111
1210 DATA 01110000
1220 DATA 00111100
1230 DATA 01110000
1240 DATA 10001000
1250 DATA 11000110
1260 RESTORE 1180: FOR F=0 TO 7
1270 READ S$:S=VAL("&b"+S$)
1280 VPOKE 8*ASC("O")+F,S
1290 NEXT F:VPOKE 8201,&B00010111
1300 DATA 00001100
1310 DATA 00001100
1320 DATA 11110010
1330 DATA 00001110
1340 DATA 00111100
1350 DATA 00001110
1360 DATA 00010001
1370 DATA 01100011
1380 FOR F=0 TO 7
1390 READ S$:S=VAL("&b"+S$)
1400 VPOKE 8*ASC("X")+F,S
1410 NEXT F:VPOKE 8203,&B11110111
1420 DATA 10000001
1430 DATA 01000010
1440 DATA 00111100
1450 DATA 00100100
1460 DATA 00100100
1470 DATA 00111100
1480 DATA 01000010
```

```
1490 DATA 10000001
1500 FOR F=0 TO 7
1510 READ S$:S=VAL("&b"+S$)
1520 VPOKE 8*ASC("*")+F,S
1530 NEXT F:VPOKE 8197,&B10000110
1540 IP=1:VPOKE 8202,&B00010111
1550 FOR F=8*ASC("0") TO 8*ASC("9")+7
1560 VPOKE F,VPEEK(F)/2
1570 NEXT F
1580 RETURN
```

ANÁLISE

O programa usa a SCREEN 1 para desenhar o tabuleiro e monitorar o jogo.

As posições dos exércitos são armazenadas na matriz A$(8,8) definida na linha 70.

Os caracteres O, X e * são redefinidos:

O é usado como exército do usuário humano e é redefinido pelas linhas de 1260 a 1290, que usam as linhas DATA de 1180 a 1250.

X é usado como exército do micro e é redefinido pelas linhas de 1380 a 1410, que usam as linhas DATA de 1300 a 1370.

***** é o caractere usado como moldura e é redefinido pelas linhas de 1500 a 1530, que usam as linhas DATA de 1420 a 1490.

Você pode redesenhar esses caracteres simplesmente alterando os dados nas linhas DATA. Experimente, por exemplo, trocar os 0's por 1's e vice-versa.

Todos os algarismos (de 0 a 9) são também redefinidos (nas linhas de 1550 a 1570) para que fiquem mais centralizados dentro da matriz 8x8 em que são definidos. Para isso basta deslocar todos os pixeis setados uma posição para a direita, dividindo o equivalente decimal da linha em que eles estão por 2!

Veja na figura 10.5 a diferença entre o 0 normal e o 0 centralizado.

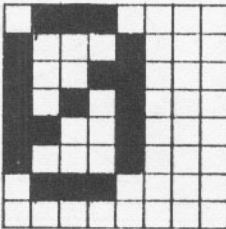
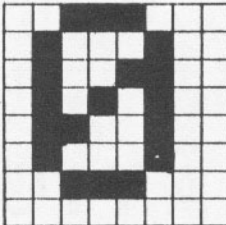



Figura 10.5 — 0 normal e 0 centralizado

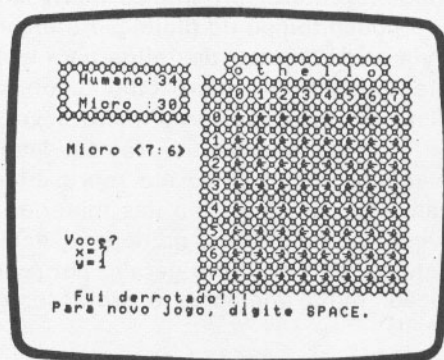
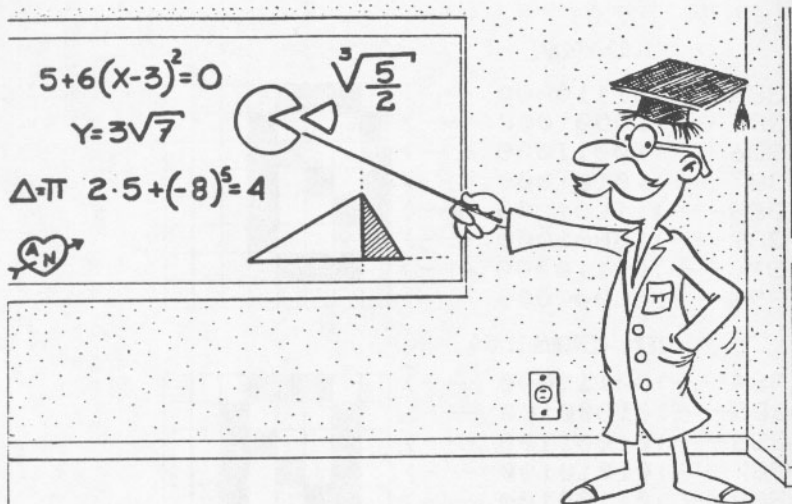
ZERO NORMAL

```
384--->01110000 --->
385--->10001000 --->
386--->10011000 --->
387--->10101000 --->
388--->11001000 --->
389--->10001000 --->
390--->01110000 --->
391--->00000000 --->
```

ZERO CENTRALIZADO

```
384--->00111000 --->
385--->01000100 --->
386--->01001100 --->
387--->01010100 --->
388--->01100100 --->
389--->01000100 --->
390--->00111000 --->
391--->00000000 --->
```

Note que o comando COLOR 4,4,4 na linha 40 faz com que o desenho da tela no vídeo não seja mostrado até que o comando COLOR da linha 170 seja executado. Isso já foi usado no programa TRON e serve apenas para mostrar a tela já pronta, de uma só vez! Experimente eliminar o COLOR da linha 40 para observar melhor a diferença que ele produz.

A estratégia do micro é definida pelas linhas DATA 870 e 880, cujos dados são armazenados na matriz C$(16,8) para que possam ser usados. Essa matriz é definida na linha 80.

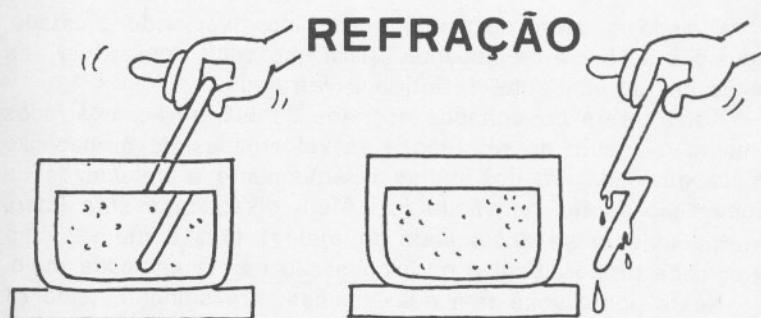
Quando o micro executa uma jogada, uma pequena melodia é executada pelas linhas de 1090 a 1120. Você pode modificá-la à vontade.

DIDÁTICOS

De todos os computadores atualmente no mercado brasileiro, os mais indicados para aplicações educacionais são os micros do padrão MSX.

Esta afirmação é facilmente explicada se avaliarmos, de um lado, os recursos de que a máquina é dotada e, do outro lado, o seu preço. O MSX apresenta a melhor relação custo/benefício quando imaginado em uso numa escola.

Nesta seção apresentamos alguns programas didáticos cuja finalidade é a de fornecer uma reduzidíssima amostra dos incríveis recursos da máquina e de suas inúmeras aplicações. O importante é que os professores percebam que, com um pouco de treinamento, podem produzir programas incríveis gastando, apesar disso, muito pouco tempo de digitação e elaboração.

Dois equívocos devem ser desfeitos para acabar com o preconceito existente contra o microcomputador na educação. O primeiro consiste em se achar que o micro deve ser utilizado na escola para se ensinar computação. Na realidade esta é, talvez, sua aplicação menos importante: ele representa, de fato, uma poderosa ferramenta para o ensino das matérias tradicionais. O segundo equívoco é o de se ficar na dependência do famigerado "software didático", normalmente gerado por programadores leigos em educação: muito melhor é o professor gerar seus próprios programas!

REFRAÇÃO

INSTRUÇÕES

Quando ocorre a passagem de luz de um meio para outro, por exemplo do ar para a água, verifica-se, de um modo geral, a mudança de intensidade da velocidade de propagação da mesma. Esta mudança de velocidade é traduzida através de um fenômeno denominado REFRAÇÃO.

Esse programa simula a passagem de um feixe de luz monocromática de um meio para outro, num dióptro plano, baseado na Lei de Snell.

De início, o programa pede que você informe os índices de refração absolutos dos dois meios. Devido aos limites físicos que a natureza impõe, os valores dos índices não devem ser inferiores a 1 e nem superiores a 5. Qualquer valor não contido no intervalo mencionado não será aceito pelo programa.

Uma vez informados os índices de refração, o programa muda a tela para o modo "SCREEN 2" e nela desenha a superfície de separação dos dois meios, a reta normal à essa superfície que passa pelo ponto de incidência da luz e a fonte luminosa, representada por um círculo verde.

A fonte luminosa é inicialmente posicionada de modo que o ângulo de incidência seja de 45 graus. No entanto, você pode mudar esse ângulo reposicionado a fonte. Para isso utilize as teclas de seta e acompanhe o valor do ângulo, que é constantemente atualizado e mostrado no box correspondente. Esse box encontra-se no quadrante inferior esquerdo da tela e contém a letra "i" referente ao ângulo de incidência.

Convém notar que o movimento da fonte só pode ser realizado sobre os lados de um quadrado imaginário, ou seja, para ângulos superiores a 45 graus funcionam apenas as teclas de seta "para cima" e "para baixo". Da mesma forma, para ângulos inferiores a 45 graus, só funcionam as teclas de seta "para esquerda" e "para direita".

Quando o valor do ângulo desejado tiver sido ajustado, pressione a "barra de espaços" para que você possa "ver" os feixes de luz incidente, refletido e refratado.

Após serem desenhados os raios de luz, serão mostrados também o ângulo de refração e as velocidades de propagação da luz em cada um dos meios relativamente à velocidade de propagação da luz no vácuo (c). Além disso, uma seta ficará indefinidamente se deslocando do meio 1 para o meio 2 com velocidade proporcional à da propagação da luz em cada meio.

Neste ponto você tem duas opções: pressionar a tecla F1 para reiniciar o programa e, desta forma, redefinir os índices de refração, ou pressionar a tecla F2 para repetir a "experiência", mantendo os mesmos índices de refração, mas tendo a oportunidade de alterar o ângulo de incidência.

Quando o índice de refração do meio 1 for maior do que o do meio 2, a pressão da tecla F3, antes que os raios de luz sejam desenhados, fará o programa mostrar o ÂNGULO LIMITE DE REFRAÇÃO.

Figura 11.1 – Programa REFRAÇÃO.

```
100 ' REFRACAO - LEI DE SNELL
110 ' LUIZ TARCISIO DE CARVALHO JR
120 '
130 ON KEY GOSUB 200,270,620
140 ON STRIG GOSUB 520
150 KEYOFF: OPEN"GRP:"FOR OUTPUT AS #1
160 SCREEN2:SPRITE$(1)=CHR$(0)+CHR$(24)+
CHR$(60)+CHR$(126)+CHR$(126)+CHR$(60)+CH
R$(24)+CHR$(0)
170 SPRITE$(2)=STRING$(3,0)+STRING$(3,25
5)+STRING$(2,0)
180 SPRITE$(3)=CHR$(124)+STRING$(3,68)+C
HR$(198)+CHR$(68)+CHR$(40)+CHR$(16)
190 SPRITE$(4)=CHR$(255)+STRING$(3,1)+CH
R$(17)+STRING$(3,1)
200 DEFUSR=&H156:Q=USR(0):KEY(1)OFF:COLO
R15,4:SCREEN0:LOCATE7,1:PRINT"REFRACAO E
M DIOPTRO PLANO":LOCATE13,3:PRINT"LEI DE
 SNELL"
210 LOCATE1,7:PRINT"Qual o indice de ref
racao do":LOCATE3,8:PRINT"meio do qual a
 luz provem?":LOCATE6,10:PRINT"Meio 1 =>
 ";:LINEINPUT X$
```

```
220 N1=VAL(X$): IF N1<1 OR N1>5 THEN LOC
ATE 15,10:PRINT SPACE$(17):GOTO 210
230 N1=INT(N1*10000+.5)/10000
240 LOCATE1,13:PRINT"Qual o indice de re
fracao do":LOCATE2,14:PRINT"meio para o
qual a luz passa?":LOCATE 6,16:PRINT"Mei
o 2 => ";:LINE INPUT Y$
250 N2=VAL(Y$):IF N2<1 OR N2>5 THEN LOCA
TE 15,16:PRINT SPACE$(17):GOTO 240
260 N2=INT(N2*10000+.5)/10000:PI=3.14159
2#:IL=100
270 KEY(2)OFF: COLOR15,1,1:SCREEN2
280 COLOR11:PSET(96,8),1:PRINT#1,"REFRAC
AO"
290 A$="LEI DE SNELL":FORY=56 TO 144 STE
P 8:PSET(16,Y),1:PRINT#1,MID$(A$,Y/8-6,1
):NEXT Y
300 LINE (24,20)-(232,188),15,B
310 A$="MEIO 1":COLOR 2:FOR Y=32 TO 72 S
TEP 8:PSET(240,Y),1:PRINT#1,MID$(A$,Y/8-
3,1):NEXT Y
320 A$="MEIO 2":COLOR3:FOR Y=120 TO 160
STEP 8:PSET(240,Y),1:PRINT#1,MID$(A$,Y/8
-14,1):NEXT Y
330 FORX=120TO128STEP8:K=X/8-13:PUT SPRI
TEK,(X,100),7,2:NEXTX
340 LINE(25,104)-(119,106),7,BF
350 LINE(136,104)-(231,106),7,BF
360 FORI=32TO174STEP4:PSET(128,I),8:PSET
(128,I+1),8:NEXTI:PUTSPRITE0,(129,95),8,
4
370 LINE(30,110)-(106,138),15,B:LINE(30,
124)-(106,124),15:LINE(30,150)-(66,178),
15,B:LINE(30,164)-(66,164),15:COLOR15:PS
ET(32,112),1:PRINT#1,"n1=":PSET(32,128),
1:PRINT#1,"n2="
380 PSET(32,152),1:PRINT#1,"i=":PSET(32,
168),1:PRINT#1,"r=":PSET(72,160),1:PRINT
#1,"(graus)"
390 PSET(48,112),1:PRINT#1,N1:PSET(48,12
8),1:PRINT#1,N2
```

```
400 X=64:Y=25:STRIG(0)ON:IFN1>N2THENKEY(
3)ON:Z=N2/N1:I1=ATN(Z/SQR(-Z*Z+1)):IL=IN
T(((I1*180)/PI)+.5)
410 T=STICK(0)
420 IFT=3ANDY=25ANDX<124THENX=X+1
430 IFT=7ANDY=25ANDX>64THENX=X-1
440 IFT=5ANDX=64ANDY<100THENY=Y+1
450 IFT=1ANDX=64ANDY>25THENY=Y-1
460 PUTSPRITE1,(X,Y),3,1
470 X1=124-X:Y1=100-Y
480 IFY1=0THENI=PI/2:IG=90:GOTO500
490 I=ATN(X1*1.25/Y1):IG=(I*180)/PI:IG=I
NT(IG+.5)
500 LINE(48,152)-(63,159),1,BF:PSET(40,1
52),1:PRINT#1,IG
510 GOTO410
520 STRIG(0)OFF:IF IG=IL THEN RG=90:GOTO
680
530 IF(SIN(I)*N1)/N2>1 OR IG>IL THEN GOT
O 720
540 Z=(SIN(I)*N1)/N2:R=ATN(Z/SQR(-Z*Z+1)
):RG=(R*180)/PI:RG=INT(RG+.5)
550 PSET(40,168),1:PRINT#1,RG
560 LINE(X+4,Y+4)-(128,104),9
570 IFIG=0 THEN LINE-(128,184),9:GOTO730
580 X2=128+60*TAN(R):Y2=104+75/TAN(R)
590 IF RG=45 THEN LINE-(188,179),9:GOTO7
00
600 IF RG<45 THEN LINE-(X2,179),9:GOTO70
0
610 IF RG>45 THEN LINE-(188,Y2),9:GOTO70
0
620 KEY(3)OFF
630 LINE(48,152)-(63,159),1,BF:PSET(40,1
52),1:PRINT#1,IL
640 IF IL=45 THEN X=64:Y=25
650 IF IL<45 THEN Y=25:X=60-(60*TAN(I1))
+64
660 IF IG>45 THEN X=64:Y=75-(75/TAN(I1))
+25
670 PUTSPRITE1,(X,Y),3,1:RG=90
```

```
680 PSET(40,168),1:COLOR15:PRINT#1,RG
690 LINE(X+4,Y+4)-(128,104),9:LINE-(188,
104),9:PSET(152,128),1:COLOR8:PRINT#1,"A
NGULO":PSET(152,144),1:PRINT#1,"LIMITE"
700 LINE(128,104)-(252-X,Y+4),6
710 GOTO730
720 LINE(X+4,Y+4)-(128,104),9:LINE-(252-
X,Y+4),9:PSET(144,128),1:COLOR8:PRINT#1,
"REFLEXAO":PSET(160,144),1:PRINT#1,"TOTA
L"
730 KEY(1)ON:KEY(2)ON:KEY(3)OFF:RETURN74
0
740 V1=INT((1/N1)*1000+.5)/1000:V2=INT((
1/N2)*1000+.5)/1000:PSET(160,22),1:COLOR
13:PRINT#1,V1;"c":PSET(160,178),1:PRINT#
1,V2;"c"
750 FORS=30TO104:PUTSPRITE10,(220,S),13,
3:FORT=1TO5*N1:NEXTT:NEXTS:FORS=105TO171
:PUTSPRITE10,(220,S),13,3:FORT=1TO5*N2:N
EXTT:NEXTS:PUTSPRITE10,(0,0),0,3:FORT=1T
O300:NEXT:GOTO750
```

ANÁLISE

As linhas 130 e 140 declaram para quais sub-rotinas deve ser desviada a execução do programa caso as teclas F1 a F3 e a barra de espaço sejam pressionadas.

As linhas 160 a 190 definem os "SPRITES" utilizados no programa. O de número 1 corresponde à fonte luminosa, o de número 2 se refere ao trecho central da superfície de separação dos dois meios, o número 3 é a seta que se desloca com velocidades proporcionais à de propagação da luz e o 4 é o símbolo de ângulo reto.

O propósito de utilizar-se "SPRITES" (2 e 4) para desenhar a região central da tela que se situa em torno do ponto de incidência da luz, é para evitar-se que o raio de luz "manche" a superfície de separação. Isso ocorre sempre que dois traçados de cores diferentes são feitos muito próximos em SCREEN 2.

As linhas 200 a 260 permitem a entrada dos índices de refração dos dois meios. Note, na linha 200, a chamada de uma sub-rotina da ROM que se inicia no endereço &H156. Sua função é "limpar" o "buffer" do teclado antes que as entradas dos índices sejam feitas.

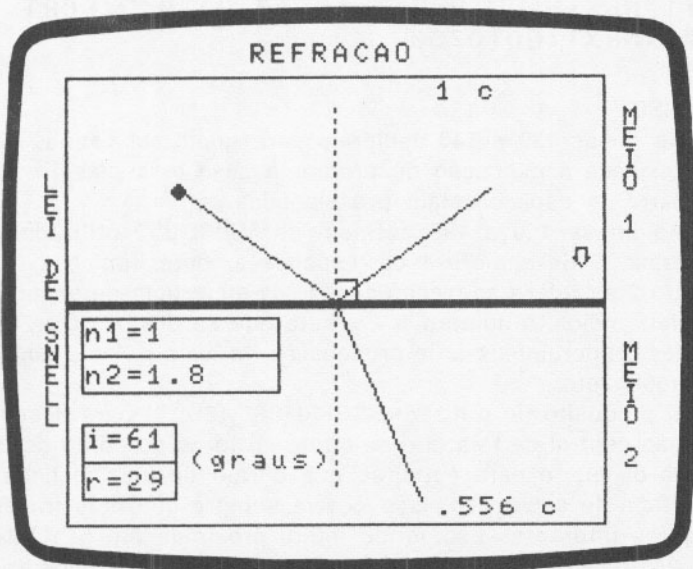
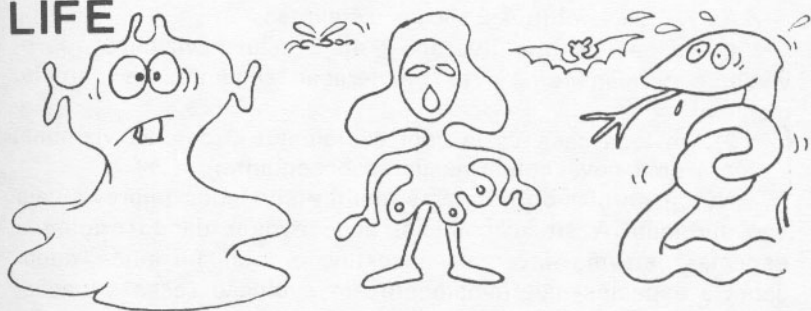
A título de experiência, elimine as instruções de chamada da sub-rotina de limpeza do "buffer" do teclado (DER USR = &H156 e Q = USR(0)) e veja o que acontece.

As linhas 270 a 480 desenham a tela e as linhas 410 a 510 permitem a movimentação da fonte luminosa e realizam o cálculo do ângulo de incidência.

As linhas 520 a 610 desenham o raio refratado e calculam o ângulo de refração. Se ocorrer reflexão total, ocorre um desvio para a linha 720.

As linhas 620 a 690 são executadas no caso de se desejar "ver" o ângulo limite, mediante a pressão da tecla F3.

As linhas 700 a 710 desenham o raio refletido; a linha 730 habilita as interrupções pelas teclas F1 e F2 e desabilita a da tecla F3. Além disso, é forçado um retorno das sub-rotinas de interrupção para a linha 740.

Finalmente, as linhas 740 e 750 calculam e mostram as velocidades de propagação da luz em cada meio e provocam o deslocamento da sala que indica essa propagação.

LIFE

INSTRUÇÕES

O programa LIFE foi adaptado de um jogo de mesmo nome criado pelo matemático inglês John H. Conway há quase duas décadas atrás.

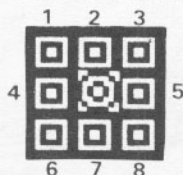
O nome LIFE (vida, em inglês) tem origem nas muitas analogias que se pode fazer entre o jogo e o desenvolvimento de culturas de bactérias (ou, de forma menos evidente, com o desenvolvimento de uma espécie qualquer em seu habitat).

Além dos interesses "biológicos" do jogo, seus aspectos matemáticos são inúmeros. Não vamos nos alongar em comentários abstratos sobre o LIFE, porém, se você estiver interessado em mais informações sobre esses aspectos, procure ler o capítulo 9 (Comunicação em Sistemas) do livro "INSTRUMENTAL PARA O PENSAMENTO" de C. H. Weddington, e os artigos de Martin Gardner (Matematical Games) das Scientific American de outubro de 1970 e fevereiro de 1971.

O princípio do programa consiste no seguinte: num tabuleiro de 24 x 32 posições (SCREEN 1), nascem se reproduzem e morrem células representadas por caracteres redefinidos).

Ao começar, o programa produz uma primeira geração de células, distribuídas ao acaso em seu habitat de 768 posições. A geração seguinte surge a partir da anterior, obedecendo a certas regras evolutivas.

Cada célula pode ter até 8 casas vizinhas com ou sem outras células (figura 12.1).

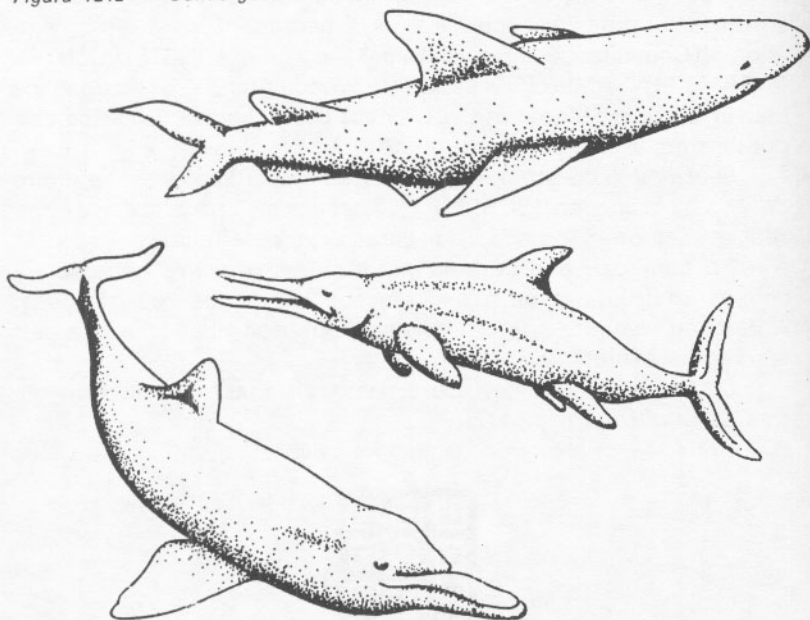
Figura 12.1 — Posições ao redor de uma célula

As regras evolutivas são as seguintes:

1) As células que tiverem 2 ou 3 células vizinhas sobrevivem e permanecem vivas na geração seguinte. As demais, morrem!

2) Em toda casa vazia com exatamente 3 células vizinhas, nascerá uma nova célula na geração seguinte.

No desenrolar do programa, muitas situações imprevisíveis vão surgindo. A situação inicial pode evoluir dando origem a espécies estáveis, levar a uma extinção total, ou ainda, quem sabe, a espécies indefinidamente em evolução (essa situação, entretanto, não é verificável empiricamente e ainda nem foi comprovada matematicamente!). Ocasionalmente surgem configurações meta-estáveis que oscilam entre duas possibilidades.

Um aspecto muito interessante a ser observado é a convergência evolutiva: como situações iniciais extremamente diferentes podem dar lugar a situações idênticas ou muito semelhantes. O tubarão, o golfinho e o ictiossauro têm origens completamente distintas: um é peixe, o segundo é mamífero e o terceiro é um réptil (figura 12.2). Mesmo assim, quando submetidos a conjuntos semelhantes de regras evolutivas do meio aquático evoluíram de modo a adquirirem formas semelhantes.

Figura 12.2 — Convergência evolutiva

Analogamente, as configurações iniciais, diferentes umas das outras, ficam submetidas às mesmas regras, simulando uma pressão de evolução, e acabam convergindo para formas similares.

A grande virtude deste programa é permitir simular, em apenas alguns minutos, um suceder de gerações que dificilmente podemos observar na prática.

Obviamente, a velocidade do programa não é muito grande. Para aumentá-la, teríamos que recorrer a linguagem de máquina ou usar um compilador BASIC.

Figura 12.3 — Programa LIFE

```
100 ' LIFE -------------------------------
110 '              Pierluigi & Renato  -
120 '---------------------------------------
140 PRINT"Digite a barra de espacos."
150 IF STRIG(0)=0 THEN BEEP : GOTO 150
160 COLOR 15,9,5 : SCREEN 1 : KEY OFF
170 ' redefine 79 e 48
180 DATA 00111100
190 DATA 01000010
200 DATA 10011001
210 DATA 10100101
220 DATA 10100101
230 DATA 10011001
240 DATA 01000010
250 DATA 00111100
260 VPOKE 8201,&B11011111
270 VPOKE 8198,&B11011111
280 FOR F%=632 TO 639
290   READ A$:A=VAL("&b"+A$)
300   VPOKE F%,A : VPOKE F%-248,A
310 NEXT F%
320 ' redefine 32 e 0
330 DATA 11111111
340 DATA 10000001
350 DATA 10111101
360 DATA 10100101
370 DATA 10100101
380 DATA 10111101
390 DATA 10000001
```

```
400 DATA 11111111
410 VPOKE 8196,&B10010110
420 VPOKE 8192,&B10010110
430 FOR F%=256 TO 263
440   READ A$:A=VAL("&b"+A$)
450   VPOKE F%,A : VPOKE F%-256,A
460 NEXT F%
470 '  sorteia celulas iniciais
480 A%=RND(-TIME)*200
490 FOR F%=1 TO A%
500   B1%=22*RND(1):B2%=3+26*RND(1)
510   VPOKE 6176+32*B1%+B2%,79
520 NEXT F%
530 '  verifica situacao e prepara
540 FOR F%=6176 TO 6880
550   C%=0
560   IF VPEEK(F%-33)<33 THEN C%=C%+1
570   IF VPEEK(F%-32)<33 THEN C%=C%+1
580   IF VPEEK(F%-31)<33 THEN C%=C%+1
590   IF VPEEK(F%-1)<33 THEN C%=C%+1
600   IF VPEEK(F%+1)<33 THEN C%=C%+1
610   IF VPEEK(F%+31)<33 THEN C%=C%+1
620   IF VPEEK(F%+32)<33 THEN C%=C%+1
630   IF VPEEK(F%+33)<33 THEN C%=C%+1
640   IF VPEEK(F%)=79 AND (C%<5 OR C%>6)
 THEN VPOKE F%,48
650   IF VPEEK(F%)=32 AND C%=5 THEN VPOK
E F%,0
660 NEXT F%
670 '  altera a tela
680 FOR F%=6144 TO 6912
690   IF VPEEK(F%)=48 THEN VPOKE F%,32
700   IF VPEEK(F%)=0 THEN VPOKE F%,79
710 NEXT F% : BEEP : BEEP : BEEP
720 GOTO 540
```

ANÁLISE

As linhas REM dividem o programa em suas principais partes. Os caracteres de códigos 79 (letra O), 48 (número 0),

0 (caractere de controle) e 32 (espaço em branco) são redefinidos e tem suas cores alteradas. Usa-se a SCREEN 1 como habitat das células.

Em linhas gerais, o programa funciona assim:

(1) Sorteiam-se as posições iniciais na tela, que inicialmente está repleta com códigos 32, e insere-se nelas o código 79 (células vivas).

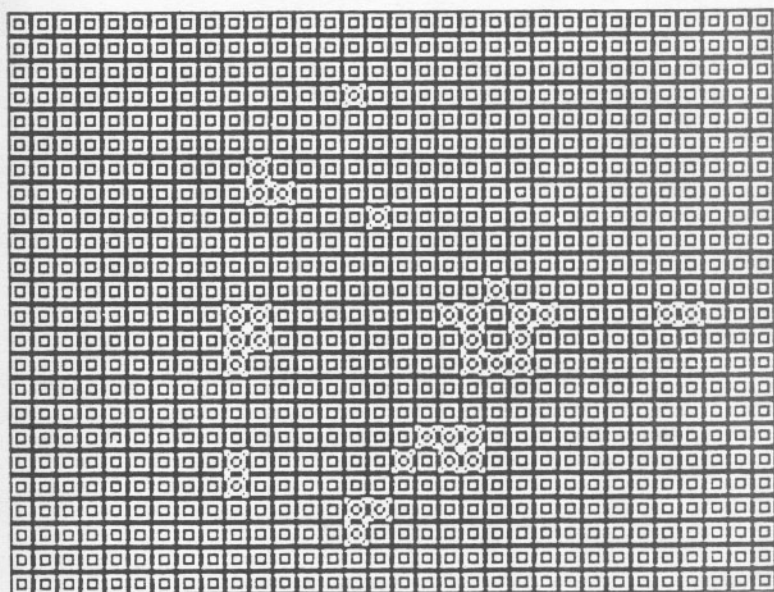
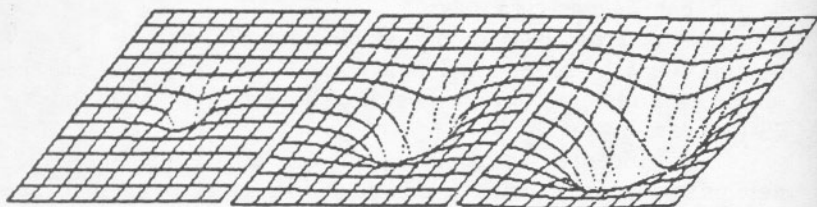
(2) Verificam-se as posições em que devem nascer novas células e insere-se nelas o código 0 (de aspecto idêntico ao do código 32).

(3) Verificam-se as posições em que as células devem morrer e insere-se nelas o código 48, cujo aspecto é idêntico ao do código 79.

(4) Verificam-se as posições com código 48 (células condenadas) e insere-se nelas o código 32. Verificam-se as posições com código 0 (embrião de célula) e insere-se nelas o código 79 (célula viva).

(5) Volta-se ao procedimento (2).

Para compreender melhor o funcionamento do programa, observe as linhas REM.

Figura 12.4 — Exemplo de tela do programa

CAMPO GRAVITACIONAL

INSTRUÇÕES

O programa que apresentamos a seguir produz no vídeo uma representação bidimensional do campo gravitacional gerado por uma massa puntiforme de acordo com a Teoria Newtoniana da gravitação. Sendo M uma massa puntiforme a ser estudada e m uma massa de prova também puntiforme, podemos escrever:

$$F = G \frac{M \times m}{d \times d} \qquad e \qquad F = m \times g$$

Onde:
F é a força de atração entre as duas massas;
d é a distância entre elas;
g é o campo gravitacional gerado por M a uma distância d:
G é a Constante Universal da Gravitação;
Sendo G e M constantes para um corpo, temos:

$$g = \frac{\text{constante}}{d \times d}$$

Isso significa que o valor g do campo gravitacional gerado por M depende apenas da distância d.

Considerando as massas num plano XY como M na origem dos eixos, pode-se escrever:

$$d^2 = X^2 + Y^2$$

Evidentemente, estamos considerando apenas o campo gra-

vitacional gerado ao longo do plano XY, ou seja, impomos a coordenada Z como sendo zero. Desta forma, temos:

$$g = g\,(d \times d)$$

Portanto:

$$g = g\,(X \times X + Y \times Y)$$

Sendo g uma função de duas variáveis, podemos então usar o plano XY para mapear as posições ao redor de M, e um eixo ortogonal Z para indicar o campo gravitacional local.

O programa inicia pedindo o valor da massa que gerará o campo gravitacional. Logo após, ele já começa a traçar o gráfico correspondente. Para parar a execução em qualquer instante, tecle CONTROL+STOP.

Experimente valores de massa entre 50 e 5000 e veja que efeitos diferentes podem ocorrer. O que acontece quando a massa é zero?

Figura 13.1 — Programa CAMPO GRAVITACIONAL

```
100 ' Campo Gravitacional
110 ' Milton Maldonado Jr.
120 '
130 INPUT"Qual a massa";M:IFM<0THEN130
140 COLOR 15,4,4:SCREEN 2:DIM A%(255):FO
R X=0 TO 255:A%(X)=200:NEXT X:BEEP
150 D=0:FOR Y=0 TO 110 STEP5:Y1=Y
160 FOR X=0 TO 130:X1=X:GOSUB 200:NEXT X
:D=D+.75
170 FOR K=Y+1 TO Y+4:Y1=K:FOR X=0 TO 130
 STEP 5:X1=X:GOSUB 200:NEXT X:D=D+.75:NE
XT K,Y
180 Y=Y+1:FOR X=0 TO 130:X1=X:GOSUB 200:
NEXT X
190 GOTO 190
200 XF=(X1-65)/10:YF=(Y1-60)/8
210 IF XF=0 AND YF=0 THEN RETURN
220 Z=M/(XF^2+YF^2):ZT=Z-Y1+180
230 IF ZT>A%(16+X1+D) THEN RETURN
240 PSET(16+X1+D,ZT):A%(16+X1+D)=ZT:RETU
RN
```

ANÁLISE

O programa simula a perspectiva cavaleira do plano distorcido pela função gravidade. As linhas 150 a 190 operam os "loops" de varredura do plano XY ao redor da massa M (situada na origem) e calculam a altura do ponto correspondente.

A sub-rotina das linhas 200 a 240 imprime o ponto nas coordenadas calculadas e coloca sua altura no elemento correspondente do valor A%. Este valor (de 256 elementos) é usado para memorizar a máxima altura impressa em cada uma das 256 colunas da tela e serve para o algorítmo de ocultação dos pontos "invisíveis' (aqueles que estão 'ocultos" por outras imagens que estão mais próximas do observador).

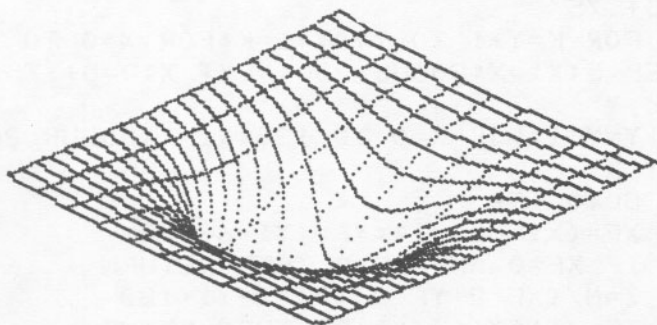
A ocultação dos pontos invisíveis foi obtida pela prioridade de impressão da frente para o fundo da imagem. A linha 230 testa se um ponto deve ou não ser impresso comparando sua altura com o elementos correspondente de A%. Se a altura do ponto for maior, faz-se a impressão. Caso contrário, não.

A função matemática propriamente dita está definida na linha 220. Experimente mudar esta função para outros polinômios, como:

$$Z = aX^2 - bY^2$$

Ou:

$$Z = a \times \text{sen}\,(bX^2 + cY^2)$$

Procure os valores a, b e c que dão os melhores resultados. É um pouco difícil, mas vale à pena!

TESTES

INSTRUÇÕES

O programa TESTE pode ter dois objetivos, o primeiro de ser um instrumento para elaboração de provas de múltipla-escolha, e o segundo como um auxiliar de estudos.

A função básica do programa é de relacionar dois tipos de itens. Para isso o programa foi dividido em duas etapas, a primeira para o estudo dos itens, a segunda para os testes envolvendo os mesmos.

Quando é selecionada a opção ESTUDAR, na tela serão mostrados dois itens que relacionam-se entre si (no nosso caso, países e capitais). Nesta etapa você deverá observar atentamente os itens para tentar memorizá-los.

Na segunda opção, TESTES, o computador elaborará uma pequena avaliação composta por 5 testes distintos, referentes aos itens estudados anteriormente.

Ao contrário dos testes normais, você deverá digitar a resposta correta por extenso, em vez de simplesmente escolher uma letra para assinalar a alternativa correta. Este artifício facilita uma maior memorização, as vezes até de maneira inconsciente. A resposta do teste deve ser digitada exatamente como ela se encontra nas alternativas.

Caso você acerte o teste, o computador irá parabenizá-lo e elaborar a próxima questão. Porém, se errar, ele fornecerá a resposta correta, para você poder se redimir de seu erro em uma próxima oportunidade. Logo após fornecer a resposta correta, o computador passará ao próximo teste.

E assim se prossegue sucessivamente até completar os 5 testes, quando o computador finalmente calculará a sua nota.

Digite o programa com cuidado. Quase todas as linhas são simples, isto é, sem multi-instruções.

Figura 14.1 — Programa TESTES

```
10 '  Testes
20 ' Carlos Eduardo Rocha Salvato
30 '
40 X=16
50 KEYOFF
60 DIMA$(X),B$(X),E(X),G(X),L(X),F%(5),R
%(5),T(5),R(5)
70 FORA=1TOX
80 READA$(A),B$(A)
90 NEXT
100 OPEN "GRP:" FOR OUTPUT AS #1
110 SCREEN2
120 P=0
130 CLS
140 COLOR2,1,1
150 PSET(80,20),1
160 PRINT#1,"Alternat ivas"
170 PSET(20,40),1
180 PRINT#1,"1--Estudar"
190 PSET(20,50),1
200 PRINT#1,"2--Testes"
210 E$=INKEY$
220 IFE$=""THENCOLOR8,1,1:PSET(20,40),1:
PRINT#1,"1":PSET(20,50),1:PRINT#1,"2":FO
RA=1TO20:NEXT:COLOR2,1,1:GOTO170
230 IFE$<>"1"ANDE$<>"2"THEN210
240 A=VAL(E$)
250 ONAGOTO260,430,210
260 CLS
270 PSET(100,20),1
280 PRINT#1,"Estudar"
290 PSET(15,40),1
300 COLOR13
310 PRINT#1,"Para prx. tem digite RETURN
"
320 LINE(192,37)-(250,49),10,B
330 COLOR8,1,1
340 FORA=1TOX
350 PSET(20,80),1
360 PRINT#1,A$(A);"        "
```

```
370 PSET(130,80),1
380 PRINT#1,B$(A);"          "
390 IFINKEY$=""THEN390
400 LINE( 0,75)-(250,90),1,BF
410 NEXT
420 GOTO110
430 FORT=1TO5
440 CLS
450 COLOR2
460 PSET(100,20),1
470 PRINT#1,"Testes"
480 ERASEG,L,R,T:DIMG(X),L(X)
490 T%=(RND(-TIME)*X)+1
500 IFE(T%)=1THEN490
510 E(T%)=1
520 G(1)=T%
530 L(T%)=1
540 FORAL=2TOX
550 G(AL)=INT(RND(-TIME)*X)+1
560 IFL(G(AL))=1THEN550
570 L(G(AL))=1
580 NEXT
590 FORZ=1TO5
600 R(Z)=INT(RND(-TIME)*5)+1
610 IFT(R(Z))=1THEN600
620 T(R(Z))=1
630 NEXT
640 PSET(20,50),1
650 PRINT#1,"Qual a capital do(a) ";A$(T
%)
660 FORRE=1TO5
670 COLOR13
680 PSET(40,(RE*15)+50),1
690 PRINT#1,B$(G(R(RE)))
700 NEXT
710 GOSUB830
720 IFR$=B$(T%)THENGOSUB1020ELSEGOSUB118
0
730 NEXT
740 SCREEN0
750 CLS
```

```
760 COLOR15
770 PRINT"Voc acertou ";P;" testes.":PRI
NT"Sua nota  ";(P*10)/5
780 LOCATE0,10,0:PRINT"Pressione <RETURN
>"
790 ERASEE:DIME(X)
800 IFINKEY$=""THEN800ELSE110
810 DATA UGANDA,KAMPALA,NIGERIA,LAGOS,EG
ITO,CAIRO,ARGELIA,ARGEL,BRASIL,BRASILIA,
ITALIA,ROMA,JAPAO,TOKIO,INGLATERRA,LONDR
ES,ARGENTINA,BUENOS AIRES,CANADA,OTTAWA,
PARAGUAI,ASSUNèAO,URUGUAI,MONTEVIDEU,VEN
EZUELA,CARACAS,CHINA,PEQUIM,AUSTRIA,VIEN
A,SUICA,BERNA
820 GOTO820
830 R$=""
840 COLOR8
850 FORA=1TO15
860 T$=INKEY$
870 IFT$=CHR$(13)THEN940
880 IFT$=""THEN860
890 IFT$=CHR$(8)THENGOTO950
900 PSET(30+(A*10),150),1
910 R$=R$+T$
920 PRINT#1,T$
930 NEXT
940 RETURN
950 A=A-1
960 COLOR1
970 PSET(30+(A*10),150),1
980 PRINT#1,RIGHT$(R$,1)
990 R$=LEFT$(R$,(LEN(R$)-1))
1000 COLOR8
1010 GOTO860
1020 CLS
1030 PSET(10,10),1
1040 P=P+1
1050 COLOR2
1060 PRINT#1,"Certo. A capital do(a) ";A
$(T%)
1070 PSET(10,30),1
```

```
1080 PRINT#1,"realmente  ";B$(T%)
1090 PSET(10,100),1
1100 COLOR2
1110 PRINT#1,"Pressione"
1120 PSET(95,100),1
1130 PRINT#1,"RETURN"
1140 PSET(95,100),1
1150 COLOR8
1160 PRINT#1,"RETURN"
1170 IFINKEY$=""THEN1090ELSERETURN
1180 CLS
1190 PSET(10,10),1
1200 COLOR2
1210 PRINT#1,"ERRADO. A capital do(a) ";
A$(T%)
1220 PSET(10,30),1
1230 PRINT#1," ";B$(T%)
1240 PSET(10,100),1
1250 COLOR2
1260 PRINT#1,"Pressione"
1270 PSET(95,100),1
1280 PRINT#1,"RETURN"
1290 PSET(95,100),1
1300 COLOR8
1310 PRINT#1,"RETURN"
1320 IFINKEY$=""THEN1240ELSERETURN
```

ANÁLISE

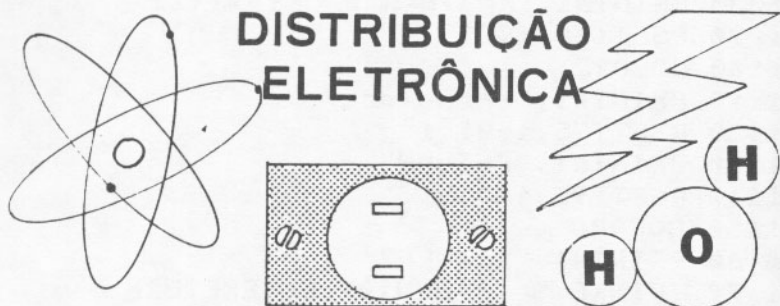
A linha 810 contém os ítens, que farão parte dos testes. Para mudar o assunto, basta alterar a linha 710, para relacionar outros ítens, como por exemplo: Símbolos e Elementos, Órgãos e Funções, Inventores e Invenções, etc.

Não se esqueça de mudar a pergunta da linha 650.

Existem ainda uma série de modificações simples e interessantes, que podem ser feitas.

Caso você queira acrescentar ou retirar algum ítem da linha 810, você deve alterar o valor da variável X na linha 40. O valor de X deve ser o número de dados da linha 810 dividido por 2.

Para aumentar ou diminuir o número de testes, você deve alterar os números nas linhas: 430 e 770.

DISTRIBUIÇÃO ELETRÔNICA

INSTRUÇÕES

Todo processo algorítmico pode ser facilmente reproduzido num programa. Atualmente, a maior ênfase educacional deve ser para a FORMAÇÃO dos alunos. Cada vez se torna menos necessário o "decorar" pois os dados são facilmente acessíveis. A função exercida pelos cálculos, sorobans, ábacos, máquinas mecânicas e também pelo lápis e papel pode ser representada com vantagens pelas pequenas calculadoras eletrônicas ou por microcomputadores ("decorar" as regras para a extração da raiz cúbica de um número envolve fundamentalmente os mesmos processos mentais, quer seja com um soroban, com lápis e papel ou com um microcomputador).

A assimilação das informações mais importantes deve ocorrer naturalmente, pelo seu uso espontâneo (não forçado).

Se quiser saber o peso atômico do ÓSMIO, pergunte a um químico profissional! Ele poderá lhe responder sem nunca ter "decorado" essa informação! Mas se quiser saber qual a distância entre as estrelas componentes do sistema CYGNUS X1, é melhor perguntar a um astrofísico!

Saber "decor" a distribuição eletrônica de um dado elemento não é nada útil para qualquer aluno (a não ser, obviamente, para fazer provas e vestibulares). Muito mais proveitoso é compreender o que significa e como se obtém a distribuição! Uma vez que isso tenha sido aprendido, obter o elétron de maior energia de um certo elemento é uma tarefa puramente mecânica!

O programa apresentado a seguir fornece a distribuição eletrônica de um elemento qualquer a partir de seu número atômico. Propositadamente, nós o desenvolvemos apenas até um estágio bem básico. Com um pouco de imaginação, você pode implementá-lo com novos dados sobre cada elementos, reproduzindo a TABELA PERIÓDICA e fazendo gráficos para verificar quais propriedades são periódicas e quais são aperiódicas.

Preste muita atenção ao digitar os dados da linha 290. Eles são fundamentais!

Figura 1.1 — Programa DISTRIBUIÇÃO ELETRÔNICA

```
10 '#   Distribuicao  Eletronica    #
20 '#   RENATO DA SILVA OLIVEIRA    #
30 '#################################
40 COLOR 15,4:SCREEN 0,,0:KEY OFF
50 LOCATE 3,5
60 INPUT"Qual o numero de eletrons";Z
70 IF Z<>INT(Z) OR Z<1 OR Z>107 THEN 50
80 CLS:C=0:PRINT,,"   Z=";Z:PRINT:PRINT
90 FOR G=1 TO 18
100 READ A$
110 N$=LEFT$(A$,1):L$=RIGHT$(A$,1)
120 I=I+1:IF I=10 THEN I=0:PRINT:PRINT
130 PRINT N$;
140 IF L$="0" THEN PRINT "s";
150 IF L$="1" THEN PRINT "p";
160 IF L$="2" THEN PRINT "d";
170 IF L$="3" THEN PRINT "f";
180 L=VAL(L$):B=0
190 FOR S=-.5 TO .5 STEP 1
200 FOR M=-L TO L:M$=STR$(M)
210 B=B+1:C=C+1
220 IF C=Z THEN 250
230 NEXT M
240 NEXT S
250 PRINT MID$(STR$(B),2);" ";
260 IF C=Z THEN 280
270 NEXT G
280 IF STRIG(0)=0 THEN 280 ELSE RUN
290 DATA 10,20,21,30,31,40,32,41,50,42,5
1,60,43,52,61,70,53,62
```

ANÁLISE

A linha 290 armazena a distribuição de LINUS PAULING (0=s, 1=p, 2=d, 3=f). A variável Z recebe o número atômico pelo teclado. O laço entre as linhas 90 e 270 se encarrega de distribuir os elétrons nas camadas, de acordo com Z e com a linha 290. O laço entre as linhas 190 e 240 é repetido duas vezes para cada sub-nível, atribuindo os SPINs aos elétrons e o laço entre as linhas 200 e 230 atribue o número quântico magnético aos elétrons.

APLICATIVOS

Existe uma característica do microcomputador pessoal que muita gente, paradoxalmente, ignora: ele pode ser ÚTIL!

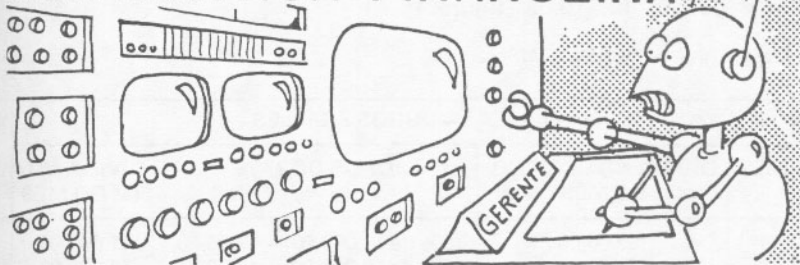
Isto significa que podemos encarregá-lo de efetuar cálculos e tarefas monótonas e repetitivas que de outra maneira nos tomariam muito tempo (um tempo às vezes tão longo que tornaria impossível executar as tarefas manualmente).

Existem alguns programas produzidos profissionalmente e tão úteis que chegam a criar "dependência" em seus usuários. A "trilogia fundamental" destes programas é constituída pelos clássicos EDITOR DE TEXTO, PLANILHA e BANCO DE DADOS. Não é nossa intenção, neste livro, apresentar aplicativos que possam competir com software deste tipo.

Ao contrário, nesta seção nosso objetivo foi o de mostrar outros tipos de programas, elaborados com a intenção de ensinar ao leitor conceitos básicos a serem usados na criação de software aplicável.

Assim, apresentamos desde um simples CALENDÁRIO que permite o cálculo das festas móveis, até o CÔNICAS, de grande utilidade para quem precisa elaborar programas que utilizem o recurso de perspectiva.

Os programas ESTATÍSTICA e MATEMÁTICA FINANCEIRA, apesar de escritos em BASIC, têm um nível semi-profissional e uma utilidade indiscutível.

MATEMÁTICA FINANCEIRA

INSTRUÇÕES

Quando uma pessoa contrata um serviço de terceiros, ela deve pagar por esse serviço. Da mesma forma, quando uma soma em dinheiro é emprestada a alguém, é razoável esperar-se um pagamento pelo uso do capital emprestado. Assim, quem empresta tem uma compensação pela perda da oportunidade de investir em outro lugar.

Chamamos de "juros" o "pagamento" pelo uso do dinheiro emprestado.

Para ilustrar, imagine que uma pessoa A empreste por um mês à uma pessoa B uma quantia de CzS 10 000,00. Suponha agora que a pessoa B, no final desse mês, pague à pessoa A o valor Cz$ 11 000,00. Podemos entender a partir deste exemplo que, a pessoa B devolveu os CzS 10 000,00 emprestados e pagou Cz$ 1 000,00 pelo uso do capital emprestado durante o mês. Os juros do mês são, portanto, de CzS 1 000,00. Note que esta quantia representa 10 por cento do capital (1 000/10 000 = 10%). Assim, dizemos que a "taxa de juros" no mês foi de 10 por cento.

Imagine que a mesma pessoa A empresta à pessoa B CzS 10 000,00, a uma taxa de juros mensal de 10 por cento durante três meses. Qual será a quantia que a pessoa B deve devolver à pessoa A, no final dos três meses?

Para responder à essa pergunta há necessidade de sabermos como são computados os juros.

Se os juros forem computados, no final de cada mês, como sendo sempre de 10 por cento sobre o "capital inicial", serão denominados JUROS SIMPLES. Assim, a quantia a ser devolvida no final de três meses será de CzS 13 000,00 (figura 16.1).

Por outro lado, se os juros forem computados, ao final de cada mês, como sendo de 10 por cento sobre a dívida no começo de cada mês, serão chamados de JUROS COMPOSTOS.

Neste caso, a quantia a ser devolvida no final dos três meses será de Cz$ 13.310,00 (figura 16.2).

Figura 16.1 — JUROS SIMPLES

TABELA I — JUROS SIMPLES			
MÊS	**DÍVIDA NO COMEÇO DO MÊS**	**JUROS DO MÊS**	**DÍVIDA NO FIM DO MÊS**
1	10.000,00	10% de 10.000,00 = 1.000,00	11.000,00
2	11.000,00	10% de 10.000,00 = 1.000,00	12.000,00
3	12.000,00	10% de 10.000,00 = 1.000,00	13.000,00

Figura 2 — JUROS COMPOSTOS — Pagamento único

TABELA II — JUROS COMPOSTOS			
PAGAMENTO ÚNICO			
MÊS	**DÍVIDA NO COMEÇO DO MÊS**	**JUROS DO MÊS**	**DÍVIDA NO FIM DO MÊS**
1	10.000,00	10% de 10.000,00 = 1.000,00	11.000,00
2	11.000,00	10% de 11.000,00 = 1.100,00	12.100,00
3	12.100,00	10% de 12.100,00 = 1.210,00	13.310,00

Note então que os "juros simples" são os juros produzidos no primeiro período (mês) multiplicados pelo número total de períodos. No exemplo, 10 por cento de 10.000,00 X 3 = 3.000,00. Os "juros compostos", entretanto, foram de 3.310,00. São maiores do que os juros simples porque computam "juros sobre juros". Em nossa economia utilizam-se quase que somente cálculos com juros compostos.

Deve ficar clara, neste ponto, a idéia de que duas somas diferentes de dinheiro podem ser equivalentes entre si, em pontos diferentes do tempo. No exemplo de juros compostos, dizemos que Cz$ 10.000,00 no dia de hoje (valor presente), equivalem a Cz$ 13.310,00 (valor futuro) daqui a três meses (número de períodos) a uma taxa de 10 por cento ao mês (taxa de juros por período).

No mesmo exemplo, a pessoa que emprestou Cz$ 10.000,00, na data de hoje, poderia concordar em receber esta quantia em pagamentos iguais no final de cada mês (série uniforme), du-

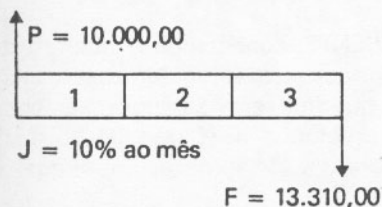
rante o período de três meses. Admitindo-se a mesma taxa de juros ao mês (10 por cento). Cz$ 10.000,00, hoje, equivalem a uma série uniforme de pagamentos de Cz$ 4.021,15 (valor da prestação) ao final de cada mês durante os três meses (figura 16.3).

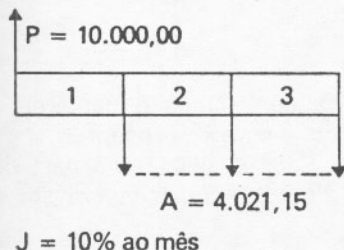
Figura 16.3 — JUROS COMPOSTOS — Série Uniforme

TABELA III — JUROS COMPOSTOS				
SÉRIE UNIFORME DE PAGAMENTOS				
MÊS	DÍVIDA NO INÍCIO DO MÊS	JUROS DO MÊS	DÍVIDA NO FIM DO MÊS	PAGAMENTO FEITO
1	10.000,00	10% de 10.000,00 = 1.000,00	11.000,00	4.021,15
2	6.978,85	10% de 6.978,85 = 697,89	7.676,74	4.021,15
3	3.655,59	10% de 3.655,59 = 365,56	4.021,15	4.021,15

Os dois planos expostos (pagamento único e série uniforme) são equivalentes e podem ser ilustrados esquematicamente pelos diagramas da figura 16.4.

Figura 16.4 — Diagramas de cálculos

"único pagamento

. . . Equivale a . . .

"série uniforme

As setas voltadas para cima representam entrada de capital e as voltadas para baixo representam saída de capital, sob o ponto de vista de quem pediu o empréstimo.

Resumindo, quantias de dinheiro não podem ser comparadas diretamente a não ser que o sejam numa mesma data. Para compará-las, em pontos diferentes do tempo, é necessário determinar os valores equivalentes através do uso de cálculos apropriados de juros compostos. Estes cálculos constituem a essência do programa MATEMÁTICA FINANCEIRA.

Assim que você fizer o programa rodar, será apresentado um "menu" inicial. Escolha a grandeza a ser calculada pressionando o número correspondente e a seguir faça a sua opção pela letra que esteja de acordo com os valores das grandezas que você conhece.

As várias opções que o programa oferece são:

1-A) Calcula o VALOR PRESENTE, conhecidos o número de períodos, a taxa de juros por período e o valor futuro.

EXEMPLO: Qual o valor que deve ser investido na data de hoje, a uma taxa de juros mensal de 1,5%, para que, daqui a 12 meses, possam ser resgatados Cz$ 100.000,00?

Entrando com a taxa de juros (1.5), o número de períodos (12) e o valor futuro (100000), o programa deve apresentar a seguinte resposta:

VALOR PRESENTE = Cz$ 83.638,74

1-B) Calcula o VALOR PRESENTE, conhecidos o número de períodos, a taxa de juros por período e o valor da prestação.

EXEMPLO: Um anúncio diz que um televisor pode ser comprado em 10 meses com uma prestação de Cz$ 1.800,00, sem entrada. Sabendo que a loja cobra 3% de juros ao mês, qual o preço do aparelho à vista?

Entrando com os valores 3, 10 e 1800 para a taxa de juros, o número de períodos e o valor da prestação, respectivamente, o programa deve apersentar a seguinte resposta:

VALOR PRESENTE = Cz$ 15.354,37

2-A) Calcula o VALOR FUTURO, conhecidos o número de períodos, a taxa de juros por período e o valor presente.

EXEMPLO: Se foram investidos Cz$ 100.000,00, na data de hoje, a uma taxa de juros mensal de 1,2%, qual deverá ser o montante daqui a um ano (12 meses)?

Entrando os valores, a resposta deve ser:

VALOR FUTURO = Cz$ 115.389,46

2-B) Calcula o VALOR FUTURO, conhecidos o número de períodos, a taxa de juros por período e o valor da prestação.

EXEMPLO: Se depositarmos Cz$ 1.500,00 todo mês numa instituição financeira que paga 2% de juros ao mês, qual deverá ser o montante a ser recebido após dois anos (24 meses) de depósitos?

O resultado deve ser:

VALOR FUTURO = Cz$ 45.632,79

3-A) Calcula o VALOR DA PRESTAÇÃO, conhecidos o número de períodos, a taxa de juros por período e o valor presente.

EXEMPLO: Qual o valor da prestação de um automóvel se ele custa Cz$ 72.000,00 à vista e sabendo-se que este valor foi financiado em 6 meses, sem entrada, a uma taxa de juros mensal de 3%?

A resposta é:

VALOR DA PRESTAÇÃO = Cz$ 13.291,02

3-B) Calcula o VALOR DA PRESTAÇÃO, conhecidos o número de períodos, a taxa de juros por período e o valor futuro.

EXEMPLO: Uma reportagem diz que um novo microcomputador será lançado daqui a 6 meses ao preço de Cz$ 15.800,00. Quanto deve ser depositado mensalmente, numa instituição financeira que paga 1,3% de juros ao mês, para que se consiga a quantia necessária para comprar o aparelho quando este for lançado?

A resposta é:

VALOR DA PRESTAÇÃO = Cz$ 2.549,04

4-A) Calcula a TAXA DE JUROS, conhecidos o valor presente, o valor da prestação e o número de períodos.

EXEMPLO: Um estabelecimento comercial financiou a quantia de Cz$ 9.000,00 em 5 meses. A prestação pedida foi de Cz$ 2.000, sem entrada. Qual a taxa mensal de juros cobrada pelo estabelecimento?

A resposta deve ser:

TAXA DE JUROS = 3.618%

4-B) Calcula a TAXA DE JUROS, conhecidos o valor pre-

sente, o valor futuro e o número de períodos.

EXEMPLO: Um terreno foi adquirido pela quantia de CzS 650.000,00, numa certa data e passou a valer Cz$ 1.210.000,00 um ano depois (12 meses). Qual foi a taxa mensal de valorização do terreno?

Fazendo o preço inicial (650000) ser o valor presente e o preço final (1210000) ser o valor futuro, onde o número de períodos é igual a 12, o programa responderá da seguinte forma:

TAXA DE JUROS = 5.315%

5-A) Calcula o NÚMERO DE PERÍODOS, conhecidos o valor presente, o valor da prestação e a taxa de juros por período.

EXEMPLO: Qual o número de prestações que uma pessoa deverá pagar para saldar um empréstimo de Cz$ 20.000,00 se ela só pode pagar no máximo a prestação de Cz$ 1.700,00 por mês, sendo que a taxa de juros mensal é de 2%?

A resposta é:

NÚMERO DE PERÍODOS = 13.55

5-B) Calcula o NÚMERO DE PERÍODOS, conhecidos o valor futuro, o valor da prestação e a taxa de juros por período.

EXEMPLO: Durante quantos meses deve-se depositar a quantia de CzS 10.000,00 para poder-se resgatar o valor de CzS 200.000,00, com uma taxa de juros de 1,8% ao mês?

O resultado fornecido pelo programa é:

NÚMERO DE PERÍODOS = 17.24

5-C) Calcula o NÚMERO DE PERÍODOS, conhecidos o valor presente, o valor futuro e a taxa de juros por período.

EXEMPLO: Quantos meses deve-se esperar para que o capital de CzS 120.000,00 triplique a uma taxa de juros mensal de 2%?

Entrando com os valores 120000, 360000 e 2 para o valor presente, o valor futuro e a taxa de juros, respectivamente, serão fornecidos pelo programa conforme segue:

NÚMERO DE PERÍODOS = 55.48

Após cada cálculo, o programa oferece invariavelmente as seguintes opções:

TECLA "N" — Para novo cálculo com a mesma opção.

TECLA "O" — Retorna ao "menu" do programa para que outra opção possa ser escolhida.

Figura 16.5 — Programa MATEMÁTICA FINANCEIRA

```
1000 '     MATEMATICA FINANCEIRA
1010 ' LUIZ TARCISIO DE CARVALHO JR
1020 '
1030 ON ERROR GOTO 2360
1040 KEYOFF: SCREEN2:COLOR15,1,1
1050 OPEN"GRP:"FOR OUTPUT AS #1
1060 GOSUB2270
1070 COLOR3:PSET(10,125):PRINT#1,"Escolh
a o valor a ser calculado"
1080 COLOR15:X$=INPUT$(1):IFASC(X$)<49OR
ASC(X$)>53THENGOSUB2260:GOTO1080
1090 O=ASC(X$):GOSUB1120
1100 LINE(10,125)-(255,135),1,BF:PSET(30
,125),1:COLOR3:PRINT#1,"Os valores conhe
cidos sao:":COLOR8:PSET(30,140),1
1110 ONOGOTO1130,1260,1390,1520,1740
1120 O=O-48:ONOGOSUB2210,2220,2230,2240,
2250:RETURN
1130 PRINT#1,"(A) [2],[4],[5]":PSET(30,1
55),1:PRINT#1,"(B) [3],[4],[5]"
1140 COLOR3:PSET(30,170),1:PRINT#1,"Esco
lha a opcao":Y$=INPUT$(1):O=ASC(Y$)-64:I
FO<1ORO>2THENGOTO1140
1150 ONOGOTO1160,1210
1160 GOSUB2300:GOSUB2010:GOSUB2030
1170 INPUT"Qual o VALOR FUTURO ";F
1180 R=F/X:GOSUB2060
1190 LOCATE12,12:PRINT"VALOR PRESENTE":L
OCATEK,14:PRINT"Cz$ ";R$
1200 GOSUB 2310:GOTO1160
1210 GOSUB2300:GOSUB2010:GOSUB2030
1220 INPUT"Qual o VALOR DA PRESTACAO ";A
1230 R=A*(X-1)/(I*X):GOSUB2060
1240 LOCATE12,12:PRINT"VALOR PRESENTE":L
OCATEK,14:PRINT"Cz$ ";R$
```

```
1250 GOSUB2310:GOTO1210
1260 PRINT#1,"(A) [1],[4],[5]":PSET(30,1
55),1:PRINT#1,"(B) [3],[4],[5]"
1270 COLOR3:PSET(30,170),1:PRINT#1,"Esco
lha a opcao":Y$=INPUT$(1):O=ASC(Y$)-64:I
FO<1ORO>2THENGOTO1270
1280 ONOGOTO1290,1340
1290 GOSUB2300:GOSUB2010:GOSUB2030
1300 INPUT"Qual o VALOR PRESENTE ";P
1310 R=P*X:GOSUB2060
1320 LOCATE13,12:PRINT"VALOR FUTURO":LOC
ATEK,14:PRINT"Cz$ ";R$
1330 GOSUB2310:GOTO1290
1340 GOSUB2300:GOSUB2010:GOSUB2030
1350 INPUT"Qual o VALOR DA PRESTACAO ";A
1360 R=A*(X-1)/I:GOSUB2060
1370 LOCATE13,12:PRINT"VALOR FUTURO":LOC
ATEK,14:PRINT"Cz$ ";R$
1380 GOSUB2310:GOTO1340
1390 PRINT#1,"(A) [1],[4],[5]":PSET(30,1
55),1:PRINT#1,"(B) [2],[4],[5]"
1400 COLOR3:PSET(30,170),1:PRINT#1,"Esco
lha a opcao":Y$=INPUT$(1):O=ASC(Y$)-64:I
FO<1ORO>2THENGOTO1400
1410 ONOGOTO1420,1470
1420 GOSUB2300:GOSUB2010:GOSUB2030
1430 INPUT"Qual o VALOR PRESENTE ";P
1440 R=P*I*X/(X-1):GOSUB2060
1450 LOCATE10,12:PRINT"VALOR DA PRESTACA
O":LOCATEK,14:PRINT"Cz$ ";R$
1460 GOSUB2310:GOTO1420
1470 GOSUB2300:GOSUB2010:GOSUB2030
1480 INPUT"Qual o VALOR FUTURO ";F
1490 R=F*I/(X-1):GOSUB2060
1500 LOCATE10,12:PRINT"VALOR DA PRESTACA
O":LOCATEK,14:PRINT"Cz$ ";R$
1510 GOSUB2310:GOTO1470
1520 PRINT#1,"(A) [1],[3],[5]":PSET(30,1
55),1:PRINT#1,"(B) [1],[2],[5]"
1530 COLOR3:PSET(30,170),1:PRINT#1,"Esco
lha a opcao":Y$=INPUT$(1):O=ASC(Y$)-64:I
FO<1ORO>2THENGOTO1530
```

```
1540 ONOGOTO1550,1660
1550 GOSUB2300:GOSUB2010
1560 LOCATE2,2:INPUT"Qual o VALOR PRESEN
TE ";P
1570 LOCATE2,4:INPUT"Qual o VALOR DA PRE
STACAO ";A
1580 LOCATE2,6:INPUT"Qual o NUM. DE PERI
ODOS ";N
1590 I=10:I2=0
1600 A1=(P*I*(1+I)^N)/((1+I)^N-1):A1=INT
(A1*100+.5)/100:I3=ABS(I-I2)/2:I2=I:IFA=
A1THEN1630
1610 IFA<A1THENI=I-I3ELSEI=I+I3
1620 GOTO1600
1630 J=INT(I*100000!+.5)/1000
1640 LOCATE10,12:PRINT"TAXA DE JUROS":LO
CATE12,14:PRINTJ;"%"
1650 GOSUB2310:GOTO1550
1660 GOSUB2300:GOSUB2010
1670 LOCATE2,2:INPUT"Qual o VALOR PRESEN
TE ";P
1680 LOCATE2,4:INPUT"Qual o VALOR FUTURO
 ";F
1690 LOCATE2,6:INPUT"Qual o NUM. DE PERI
ODOS ";N
1700 I=(F/P)^(1/N)-1
1710 J=INT(I*100000!+.5)/1000
1720 LOCATE10,12:PRINT"TAXA DE JUROS":LO
CATE12,14:PRINTJ;"%"
1730 GOSUB2310:GOTO1660
1740 PRINT#1,"(A) [1],[3],[4]":PSET(30,1
55),1:PRINT#1,"(B) [2],[3],[4]":PSET(30,
170),1:PRINT#1,"(C) [1],[2],[4]"
1750 COLOR3:PSET(30,185),1:PRINT#1,"Esco
lha a opcao":Y$=INPUT$(1):O=ASC(Y$)-64:I
FO<1ORO>3THENGOTO1750
1760 ONOGOTO1770,1850,1930
1770 GOSUB2300:GOSUB2010
1780 LOCATE2,2:INPUT"Qual o VALOR PRESEN
TE ";P
1790 LOCATE2,4:INPUT"Qual o VALOR DA PRE
STACAO ";A
```

```
1800 LOCATE2,6:PRINT"Qual o taxa de juro
s":LOCATE3,7:INPUT"expressa em porcentag
em";J
1810 I=J/100:N=(LOG(A)-LOG(I)-LOG(A/I-P)
)/LOG(1+I)
1820 N=INT(N*100+.5)/100
1830 LOCATE8,12:PRINT"NUMERO DE PERIODOS
":LOCATE12,14:PRINTN
1840 GOSUB2310:GOTO1770
1850 GOSUB2300:GOSUB2010
1860 LOCATE2,2:INPUT"Qual o VALOR FUTURO
 ";F
1870 LOCATE2,4:INPUT"Qual o VALOR DA PRE
STACAO ";A
1880 LOCATE2,6:PRINT"Qual o taxa de juro
s":LOCATE3,7:INPUT"expressa em porcentag
em";J
1890 I=J/100:N=LOG(F*I/A+1)/LOG(I+1)
1900 N=INT(N*100+.5)/100
1910 LOCATE8,12:PRINT"NUMERO DE PERIODOS
":LOCATE12,14:PRINTN
1920 GOSUB2310:GOTO1850
1930 GOSUB2300:GOSUB2010
1940 LOCATE2,2:INPUT"Qual o VALOR PRESEN
TE ";P
1950 LOCATE2,4:INPUT"Qual o VALOR FUTURO
 ";F
1960 LOCATE2,6:PRINT"Qual a taxa de juro
s":LOCATE3,7:INPUT"expressa em porcentag
em";J
1970 I=J/100:N=LOG(F/P)/LOG(1+I)
1980 N=INT(N*100+.5)/100
1990 LOCATE8,12:PRINT"NUMERO DE PERIODOS
":LOCATE12,14:PRINTN
2000 GOSUB2310:GOTO1930
2010 PRINTSTRING$(39,219):LOCATE0,22:PRI
NTSTRING$(39,219);
2020 RETURN
2030 LOCATE2,2:PRINT"Qual a taxa de juro
s":LOCATE3,3:INPUT"expressa em porcentag
em";J
```

```
2040 LOCATE2,5:INPUT"Qual o numero de pe
riodos";N
2050 I=J/100:X=(1+I)^N:LOCATE2,7:RETURN
2060 R=INT(R*100+.5)/100
2070 R$=STR$(R):K=LEN(R$):FORU=2TOK-1:IF
MID$(R$,U,1)="E"ORMID$(R$,U,1)="D"THENRE
TURN
2080 NEXTU
2090 C$="":IFR=INT(R)THENGOTO2130
2100 R$=STR$(R):IFMID$(R$,LEN(R$)-1,1)="
."THENC$=","+RIGHT$(R$,1)+"0":GOTO2120
2110 C$=","+RIGHT$(R$,2)
2120 R=INT(R):GOTO2140
2130 C$=",00"
2140 R$=STR$(R):R$=MID$(R$,2):H$=""
2150 K=LEN(R$):IFK<=3THENH$=R$:GOTO2190
2160 NP=INT(K/3):RE=K-NP*3
2170 IFRE=0THENFORSS=1TOK-3STEP3:FORS=SS
TOSS+2:H$=H$+MID$(R$,S,1):NEXTS:H$=H$+".
":NEXTSS:H$=H$+RIGHT$(R$,3):GOTO2190
2180 H$=LEFT$(R$,RE):FORSS=RE+1TOKSTEP3:
H$=H$+".":FORS=SSTOSS+2:H$=H$+MID$(R$,S,
1):NEXTS:NEXTSS
2190 IFR=0THENH$="0"
2200 R$=H$+C$:K=LEN(R$):K=(35-K)\2:RETUR
N
2210 PSET(30,35),1:PRINT#1,"[1] VALOR PR
ESENTE":RETURN
2220 PSET(30,50),1:PRINT#1,"[2] VALOR FU
TURO":RETURN
2230 PSET(30,65),1:PRINT#1,"[3] VALOR DA
 PRESTACAO":RETURN
2240 PSET(30,80),1:PRINT#1,"[4] TAXA DE
JUROS":RETURN
2250 PSET(30,95),1:PRINT#1,"[5] NUMERO D
E PERIODOS":RETURN
2260 PSET(70,150),1:PRINT#1,"OPCAO INVAL
IDA":FORI=1TO800:NEXTI:LINE(70,150)-(190
,160),1,BF:RETURN
2270 CLS:PSET(40,5),1:COLOR8:PRINT#1,"MA
TEMATICA FINANCEIRA"
```

```
2280 COLOR7:GOSUB2210:GOSUB2220:GOSUB223
0:GOSUB2240:GOSUB2250
2290 RETURN
2300 COLOR15,4:SCREEN0:WIDTH39:RETURN
2310 LOCATE0,20:PRINT"'O' P/OUTRA OPCAO
OU 'N' P/NOVO CALCULO"
2320 A$=INKEY$
2330 IFA$="O"ORA$="o"THENRUN
2340 IFA$="n"ORA$="N"THENRETURN
2350 GOTO2320
2360 COLOR15,8:CLS:LOCATE10,9:PRINT"VALO
R(ES) INVALIDO(S)":FORU=1TO1000:NEXTU:RU
N
```

ANÁLISE

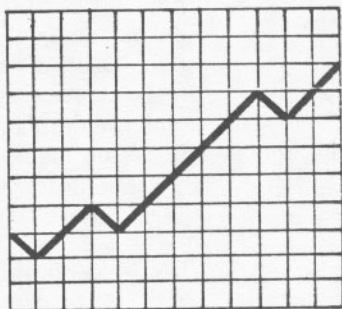
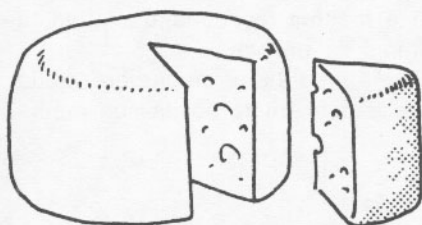
O programa não contém detalhes muito complicados em sua estrutura. Envolve apenas as fórmulas de cálculo de juros compostos, conhecidas por quem já estudou alguma vez MATEMÁTICA FINANCEIRA, e certas subrotinas de formatação da tela e de tratamento dos valores envolvidos nos cálculos.

As linhas 1030 a 1110 constroem a tela correspondente ao 'menu, e permitem a escolha da opção de cálculo desejada.

Os cálculos de cada uma das opções estão contidos nas linhas do programa que se seguem:

Opção 1-A: linhas 1160 a 1200.
Opção 1-B: linhas 1210 a 1250.
Opção 2-A: linhas 1290 a 1330.
Opção 2-B: linhas 1340 a 1380.
Opção 3-A: linhas 1420 a 1460.
Opção 3-B: linhas 1470 a 1510.
Opção 4-A: linhas 1550 a 1650.
Opção 4-B: linhas 1660 a 1730.
Opção 5-A: linhas 1770 a 1840.
Opção 5-B: linhas 1850 a 1920.
Opção 5-C: linhas 1930 a 2000.

Além disso, é interesse notar a estrutura de funcionamento da sub-rotina que se inicia na linha 2060 e termina na linha 2200. Ela converte a variável numérica (R), correspondente às respostas, em uma variável "string" (R$), formatada segundo o padrão financeiro. Por exemplo, o valor 23456.4587, armazenado em R, torna-se "23.456,46", armazenado em R$.

ESTATÍSTICA

INSTRUÇÕES

Frequentemente precisamos analisar as características de um certo grupo de dados, como o comportamento da variável nota em uma classe ou a distribuição do salário em uma certa empresa.

Para podermos ter uma visualização mais completa do conjunto dos dados, lançamos mão de certos gráficos, mais explicitamente o histograma e o gráfico em setores. Histogramas são gráficos de barras (fig. 17.2a), cujo eixo das abcissas representa as classes (ou partições) e isso quer dizer que os dados representados por aquela barra encontram-se entre os valores extremos que compreendem a barra. No histograma, dependendo do número de classes e dos dados a serem tratados, todas as classes poderão ter uma base maior ou menor. O eixo das ordenadas representa a frequência relativa, que nada mais é do que a porcentagem de cada uma das classes.

O outro gráfico apresentado pelo programa é o gráfico em setores, que mostra qual a porcentagem de cada classe em relação ao conjunto total dos dados.

A figura 17.2b mostra um gráfico em setores.

O programa também calcula algumas medidas de tendência central e de dispersão

MÉDIA A média aritmética é a soma de todos os valores da amostra dividido pelo número de dados da amostra.

$ME = \Sigma xi/n$

MODA A moda é definida como a realização mais frequente do conjunto de valores observados. Como estamos trabalhando em um histograma, resolve-

mos optar como sendo moda a média aritmética de classe que apresenta a maior frequência relativa. Caso o conjunto de dados tenha duas ou mais classes com a mesma frequência relativa, a moda poderá ter mais de um valor.

DESVIO MÉDIO — O desvio médio é uma medida de dispersão: mede a concentração dos dados em torno da sua média portanto,

DM = ΣI(xi-ME)I/n

VARIÂNCIA — A variância é outra medida de dispersão e é definida como:

VAR = Σ(xi-ME) 2/n

DESVIO PADRÃO — Como a variância expressa o desvio quadrático médio, é definido o DESVIO PADRÃO, que é a raiz quadrada da variância.

DP = SQR(VAR)

Deste modo a nossa medida de dispersão ficará expressa na mesma unidade que o conjunto dos dados.

Figura 17.1 — Programa ESTATISTICA

```
10 '** HISTOGRAMA E GRAFICO SETORIAL **
20 '* Henrique de Figueredo Luz -1986 *
30 '***********************************
40 MAXFILES=2:OPEN"GRP:"AS#1:POKE&HFCAB,
255:DIM D(10),P(10),A$(10):DEFUSR=&H156:
KEY OFF
50 KEY(1) ON: KEY(2) ON: KEY(3) ON:KEY(4
) ON:WIDTH 40:
60 ON ERROR GOTO 1310: SCREEN0:ON KEY GO
SUB 150,520,720,1340:CLS
70 COLOR,4:PRINT"************ Estatistic
a **************"
80 PRINT"[F1].....Entrada de dados"
90 PRINT"[F2].....Histograma"
100 PRINT"[F3].....Grafico em setores"
110 PRINT"[F4].....Imprime o video"
120 PRINT,,"Selecione a opao desejada"
```

```
130 A=USR(0):GOTO 130
140 '******* I/O de dados *************
150 CLS
160 PRINT"*********** Entrada de dados *
*********":PRINT:PRINT
170 LOCATE10: PRINT"1.....Le do cassete"
:LOCATE10: PRINT"2.....Grava no cassete"
:LOCATE10: PRINT"3.....Entrada via tecla
do":LOCATE10: PRINT"4.....Alteracao de d
ados":PRINT:PRINT::LOCATE10:PRINT"Outra
tecla retorna ao menu"
180 A$=INPUT$(1)
190 IF A$="4" THEN GOTO 390
200 IF A$="3" THEN GOTO 360
210 IF A$="2" THEN GOTO 300
220 IF A$<>"1" THEN CLS:GOTO 50
230 CLS:INPUT"Qual o nome do arquivo";A$
: A$="CAS:"+A$
240 OPENA$ FOR INPUT AS#2
250 INPUT #2,ND,NP
260 ERASED: DIM D(ND-1)
270 FOR F=0 TO ND-1
280 INPUT#2,D(F):NEXT:CLOSE#2
290 GOTO 150
300 CLS:INPUT"Qual o nome do arquivo";A$
: A$="CAS:"+A$
310 OPENA$ FOR OUTPUT AS#2
320 PRINT#2,ND,NP
330 FOR F=0 TO ND-1
340 PRINT#2,D(F):NEXT:CLOSE#2
350 GOTO 150
360 CLS:INPUT"Entre o numero de dados";N
D:INPUT"Entre o numero de particoes (mx
256)";NP:D1=ND-1:ERASE D:DIMD(D1):S=0
370 PRINT"No.       DADO"
380 FOR F=0 TO D1: PRINTF+1:LOCATE10,CSR
LIN-1:LINE INPUTDA$:D(F)=VAL(DADO$): NEX
T F: GOTO 150
390 D1=ND-1
400 CLS: PRINT"****************Alteracao*
**************":LOCATE10: PRINT"1.....Nu
mero de dados":LOCATE10: PRINT"2.....Num
ero de particoes":LOCATE10: PRINT"3.....
Dados"
```

```
410 A$=INPUT$(1):P=ASC(A$):IF P<49 OR P>
51 THEN GOTO 150
420 IF P=49 THEN PRINT"valor atual"ND:LI
NEINPUT"entre o novo valor";DADOS$:ND=VA
L(DADOS$):GOTO 440
430 IF P=50 THEN PRINT"valor atual"NP:LI
NEINPUT"entre o novo valor";DADOS$:NP=VA
L(DADOS$):GOTO 400
440 CLS:PRINT"Precione:":PRINT" [RETURN]
 para alterear":PRINT" barra para contin
uar":DIMQ(ND-1):FOR F=0 TO D1:Q(F)=D(F):
NEXT:ERASE D
450 FOR F=1 TO ND
460 LOCATE5,5:PRINT"dado no"F"valor="Q(F
-1)"              ": A$=INPUT$(1)
470 IF A$=CHR$(13) THEN LOCATE21,5:LINE
INPUTDA$:Q(F-1)=VAL(DA$)
480 NEXT
490 DIMD(ND-1):FOR F=0 TO ND-1:D(F)=Q(F)
:NEXT:ERASE Q
500 GOTO 400
510 '********** monta histograma*******
520 GOSUB 980: SCREEN2: RESTORE: DRAW"BM
7,0D184R248":PRESET(20,185):PRINT#1,"F r
etorna ao menu"
530 FOR F=0 TO7:READA:B$=B$+CHR$(A):NEXT
:SPRITE$(1)=B$:B$=""
540 FOR F=0 TO7:READA:B$=B$+CHR$(A):NEXT
:SPRITE$(2)=B$
550 LINE(0,0)-(256,8),13,BF:PRESET(0,0):
 FOR F=0 TO 4: READ A$: PRESET(6*F,0),13
: PRINT#1,A$: NEXT
560 FOR F=0 TO 4: READ A$: PRESET(123+6*
F,0),13: PRINT#1,A$: NEXT
570 I=248/NP
580 FOR F=0 TO NP-1
590 LINE(7+F*I,183)-(7+I*(F+1),183-P(F,2
)),14,B: NEXTF
600 PUTSPRITE2,((VER-1)*I+I,179-P(VER,2)
),3
610 PUTSPRITE1,(5+(VER-1)*I+I,184),3
```

```
620 A$=INPUT$(1)
630 P=ASC(A$)
640 IF P=31 AND VER<NP THEN VER=VER+1
650 IF P=30 AND VER>0 THEN VER=VER-1
660 LINE(30,0)-(119,7),13,BF: LINE(153,0
)-(256,8),13,BF: PRESET(30,0),13
670 A$=STR$(CSNG(MI+((MA-MI)/NP*VER)))+C
HR$(255)+STR$(CSNG(P(VER,1)/ND))
680 L=36: Q=0: FOR F=1 TO LEN(A$): B$=MI
D$(A$,F,1):IF B$=CHR$(255) THENL=153: Q=
F ELSE PRINT#1,B$: PRESET(L+6*(F-Q),0),1
3
690 NEXT
700 IF P=70 THEN RETURN 60 ELSE 600
710 '********* grafico em setores *****
720 GOSUB 980:CLS:LR=50:SR=44
730 INPUT"Deseja colocar a frequencia na
s          particoes";A$:
740 SCREEN2:A2=0:GC=1
750 FOR F=0 TO NP-1
760 IF P(F,1)=0 THEN GOTO 920
770 GC=GC+2:IF GC=15 THEN GC=2
780 IF GC=16 THEN GC=3
790 A1=A2:A2=A2+P(F,1)/ND*2*3.1415926#
800 AA=(A1+A2)/2
810 CX=130+COS(AA)*(LR-SR)
820 CY=80-SIN(AA)*(LR-SR)
830 CIRCLE(CX,CY),SR,15,-A1-1E-03,-A2,1
840 LY=CY-SIN(AA)*(SR+20)-5
850 IF A$="N" THEN GOTO 920
860 IF COS(AA)>=0THEN LX=183ELSE LX=0
870 PRESET(LX,LY):PRINT#1,USING"##.###";
(100*P(F,1)/ND);:PRINT#1,"%"
880 LINE (LX,LY+10)-(LX+64,LY+10),15
890 IF COS(AA)<0 THEN LX=LX+64
900 LINE(CX+COS(AA)*.8*SR,CY-SIN(AA)*.8*
SR)-(CX+COS(AA)*1.1*SR,CY-SIN(AA)*1.1*SR
)
910 LINE(CX+COS(AA)*1.1*SR,CY-SIN(AA)*1.
1*SR)-(LX,LY+10)
920 NEXT F
```

```
930 PRESET (20,180)
940 PRINT#1,"F retorna ao menu ";
950 A$=INPUT$(1)
960 IF A$="F" THEN: RETURN 60 ELSE 930
970 '*****tratamento dos dados *******
980 PRINT:PRINT"*************** CALCULAND
O **************"
990 S=0:FOR F=0 TO ND-1
1000 IF F=0 THEN MA=D(F):MI=MA
1010 IFMI>D(F) THEN MI=D(F)
1020 IFMA<D(F) THEN MA=D(F)
1030 S=S+D(F):NEXT F:
1040 ME=S/ND: D1=ND-1:ERASEP:DIMP(NP,2)
1050 FOR F=0 TO D1
1060 IF MA-MI<>0 THEN X=INT((1.5+(D(F)-M
I)*(NP-1)/(MA-MI)))-1
1070 P(X,1)=P(X,1)+1:NEXT
1080 M=0: W=0
1090 FOR F=0 TO NP
1100 IF P(M,1)<P(F,1) THEN M=F
1110 NEXT
1120 PM=P(M,1):CLS
1130 PRINT"MINIMO"MI: PRINT"MAXIMO"MA:PR
INT"MEDIA"ME
1140 FOR F=0 TO NP-1: IF P(F,1)=PM THEN
 PRINT"MODA"MI+(MA-MI)/NP*(F+.5)
1150 NEXT
1160 DMA=0: VAR=0
1170 FOR F=0 TO D1
1180 DMA=DMA+ABS(D(F)-ME)
1190 S1=D(F)-ME
1200 VAR=VAR+S1^2
1210 NEXT
1220 PRINT"Desvio medio"DMA/ND
1230 PRINT"Variancia"VAR/(ND)
1240 PRINT"Desvio padrao";SQR(VAR/(ND))
1250 A$=INPUT$(1)
1260 FOR F=0 TO NP
1270 P(F,2)=INT(.5+174*P(F,1)/PM)
1280 NEXT F
1290 RETURN
```

```
1300 '********* rotina de erro ********
1310 CLS: LOCATE10,12:PRINT"********ERRO
*********":FOR F=0 TO300: NEXT
1320 RESUME 50
1330 '********* copia de tela**********
1340 IF PEEK(&HFCAF)=2THEN GOTO 50000
1350 FOR Q=0 TO 23:FOR I=0 TO 39:LPRINTC
HR$(VPEEK(I+40*Q));:NEXTI:LPRINT CHR$(&H
A);CHR$(&HD);:NEXTQ:RETURN
1360 DATA 16,56,124,254,56,56,56,56
1370 DATA 8,12,254,255,254,12,8,0
1380 DATA P,A,R,T,=, ,r,e,l,=
50000 RETURN
```

ANÁLISE

O programa possui 4 blocos principais:

* Entrada de dados
* Montagem do histograma
* Gráfico em setores
* Cópia de tela

ENTRADA DE DADOS — linhas 150-500

O bloco gerencia a entrada via teclado, leitura, gravação e alteração do conjunto de dados que estão na matriz D. A leitura do arquivo é feita nas linhas 230 a 280, e a gravação do arquivo é feita nas linhas 300 a 340.

Os dados são gravados e lidos do cassete na seguinte ordem:

ND — Número de Dados
NP — Número de Participações
D(f)— elementos da matriz de Dados

A entrada dos dados via teclado é feita nas linhas de 360-380 e tem um funcionamento bastante simples.

As alterações são feitas a partir da linha 400, indo até a linha 500 onde termina o bloço.

HISTOGRAMA — linhas de 510-700

A tela é montada entre as linhas 520-590, e as barras do histograma são feitas pelo comando LINE no loop das linhas 580-590. O restante do bloco (600-700) controlam os "SPRITES".

movidos pelas teclas de controle do cursor, indicando a participação e a frequência relativa, além de imprimir os respectivos valores (com tabulação de SCREEN 0 — linha 670-680) no topo da tela.

GRÁFICO EM SETORES — linhas de 720-950

Os setores do gráfico em setores são montados pelo comando CIRCLE (linha 830), A variável A1 indica o ângulo de início da partição e a variável A2, o ângulo final da partição. As linhas de 860 a 920 imprimem a frequência relativa das classes (LX e LY são as coordenadas de referência para a impressão).

TRATAMENTO DOS DADOS — linhas de 980 a 1280

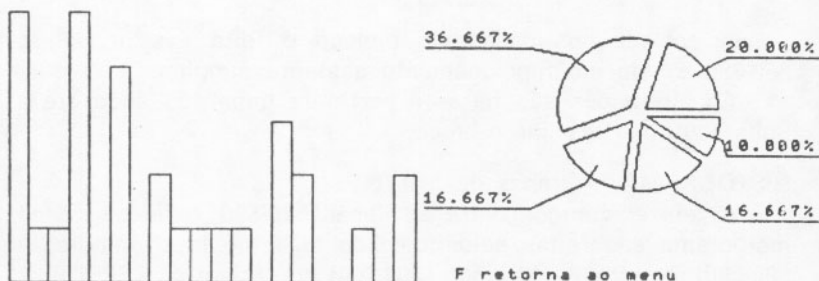
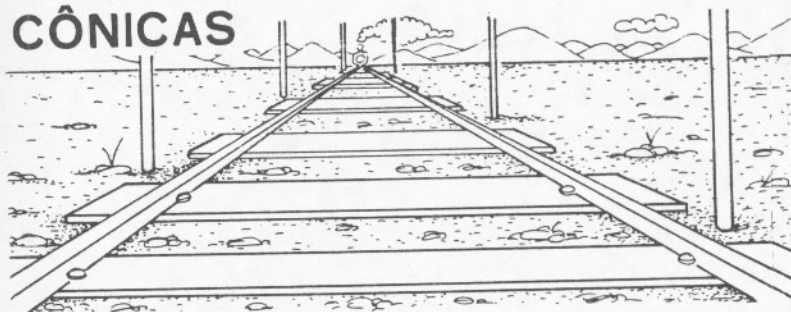
As linhas de 1000 a 1030 calculam a soma dos dados e acham o maior dado e o menor dado, variável MA e MI respectivamente.

O loop das linhas 1050-1070 calcula quantos dados estão compreendidos em cada uma das classes colocando-os na primeira coluna da matriz P, e o loop das linhas 1090-1110 calcula qual a maior classe.

As linhas de 1130 a 1280 calculam e apresentam as características do conjunto de dados.

CÓPIA NA IMPRESSORA — linhas de 1340 a 1350

Sempre que a tecla F4 é pressionada, o programa copia o que está na tela na impressora. Esta rotina começa na linha 1340, mas só copia os dados que estão na SCREEN 0. Se você possui uma impressora gráfica e que aproveitar o potencial dela com este programa, realize um MERGE com a rotina CÓPIA GRÁFICA (listada na pág. 138). Para realizar as suas cópias na impressora basta pressionar F4 e uma outra tecla qualquer.

Figura 17.2 — Histograma (a) e Gráfico em Setores (b)

CÔNICAS

INSTRUÇÕES

Este programa permite produzir a perspectiva de um sólido geométrico definido pelo usuário, como uma observação real de um objeto.

Ao ser chamado, ele apresenta um "menu" com as seguintes opções:

1. ENTRAR DADOS: permite entrar as coordenadas dos vértices e ar respectivas arestas. O programa pede inicialmente o número de vértices e arestas que compõem o sólido e, em seguida, os seus respectivos valores.

2. OBSERVADOR: pede o local do observador, isto é, as coordenadas em relação à origem da base ortogonal. A seguir, ele pede a distância do plano de projeção (ou plano de quadro) em relação à origem.

4. VISUALIZAR: imprime a vista perspectivada na tela do vídeo.

5. CORRIGIR: permite redigitar as coordenadas de um ou mais vértices do sólido. O número do ponto a ser corrigido é pedido e, em seguida, o programa pergunta se deve confirmar a correção. A qualquer instante pode-se mudar de ponto utilizando-se as teclas "cursor para a direita" e "cursor para a esquerda".

6. SALVAR DADOS: salva em disco ou fita as informações sobre o sólido, observador e distância do plano de quadro.

7. LER DADOS: lê as mesmas informações do item 6 do disco ou fita.

Experimente o programa para estes valores:

Número de vértices = 5
Número de arestas = 8
Vértices:

1: 5,5,0
2: 5,-5,0
3: -5,-5,0
4: -5,5,0
5: 0,0,6

Arestas:

1: 1,2
2: 2,3
3: 3,4
4: 4,1
5: 1,5
6: 2,5
7: 3,5
8: 4,5

Coordenadas do observador: 10,12,20
Distância do plano: 0

Se durante algum cálculo surgir a mensagem "EXCEDEU LIMITES DO VÍDEO", iso significa que algum ponto projetado está fora da tela. Nesse caso, tente reduzir o "zoom", colocando uma distância positiva para o plano de quadro.

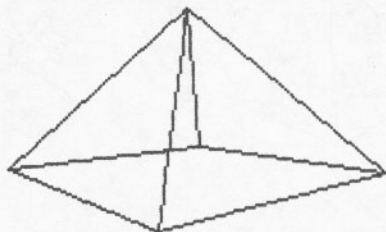
A digitação do programa não requer maiores cuidados. Para iniciar a execução digite RUN, mas após isto não digite outro RUN para não apagar as variáveis. Se, em algum instante, o programa for interrompido e você deseja preservar as variáveis, digite GOTO 410.

Figura 18.1 — Programa PERSPECTIVAS CÔNICAS

```
10 REM PERSPECTIVAS CONICAS
20 REM MILTON MALDONADO JR.
30 FL=0:GOTO 410
40 CLEAR:GOSUB 400
50 CLS:INPUT "QUANTOS VERTICES ";N:INPUT
 "QUANTAS ARESTAS ";M
60 CLS:PRINT "ENTRE AS COORDS. DOS VERTS
 (X,Y,Z)"
70 FOR P=1 TO N:PRINT "VERTICE ";P:INPUT
 X(P),Y(P),Z(P):NEXT P
80 CLS:PRINT "ENTRE ARESTAS (INICIO, FIM
)"
```

```
90 FOR P=1 TO M:PRINT "ARESTA ";P;:INPUT
 I(P),J(P):NEXT P
100 GOTO 410
110 CLS:INPUT "COORDS. OBSERVADOR (X,Y,Z
)";OX,OY,OZ:IF OX=0 AND OY=0 AND OZ=0 TH
EN 110
120 INPUT "DIST. DO PLANO PROJETANTE ";R
:RETURN
130 CLS:A=SQR(OX*OX+OY*OY+OZ*OZ):D1=A-R:
K=SQR(OX*OX+OY*OY)
140 B=-OY/K:C=OX/K:D=0:E=-OX*OZ/A/K
150 F=-OY*OZ/A/K:G=(OX*OX+OY*OY)/A/K
160 H=OX/A:I=OY/A:J=OZ/A
170 FOR P=1 TO N:T=(R/A-1)*A*A/(OX*(X(P)
-OX)+OY*(Y(P)-OY)+OZ*(Z(P)-OZ))
180 A(P)=T*(X(P)-OX)+OX:B(P)=T*(Y(P)-OY)
+OY:C(P)=T*(Z(P)-OZ)+OZ:D(P)=129+10*(B*A
(P)+C*B(P)+D*C(P))
190 E(P)=96-10*(E*A(P)+F*B(P)+G*C(P)):F(
P)=H*A(P)+I*B(P)+J*C(P)
200 IF D(P)>255 OR E(P)>191 OR D(P)<0 OR
 E(P)<0 THEN 220
210 NEXT P:FL=0:RETURN
220 FL=1:PRINT "EXCEDEU LIMITES DO VIDEO
":FOR P=1 TO 1000:NEXT P:RETURN
230 IF FL=1 THEN 220
240 SCREEN 2:FOR P=1 TO M:LINE(D(I(P)),E
(I(P)))-(D(J(P)),E(J(P))):NEXT P:A$=INPU
T$(1):SCREEN 0:RETURN
250 CLS:INPUT "PONTO A CORRIGIR (0=RETOR
NA) ";P:IF P<1 OR P>N THEN RETURN
260 CLS:PRINT "PONTO ";P:PRINT "X=";X(P)
;"  Y=";Y(P);"  Z=";Z(P)
270 PRINT:PRINT:PRINT "DESEJA CORRIGIR (
S,N,<,>)":A$=INPUT$(1):IF A$="N" THEN RE
TURN
280 IF A$=CHR$(28) THEN P=P+1:IF P>N THE
N P=N
290 IF A$=CHR$(29) THEN P=P-1:IF P<1 THE
N P=1
300 IF A$<>"S" AND A$<>"s" THEN 260
```

```
310 PRINT:INPUT "ENTRE NOVAS COORDS (X,Y
,Z) ";X(P),Y(P),Z(P):GOTO 260
320 CLS:INPUT "NOME DO ARQUIVO ";N$:GOSU
B 470:PRINT "CONFIRMA GRAVACAO (S,N) ?":
A$=INPUT$(1):IF A$<>"S" AND A$<>"s" THEN
 RETURN
330 IF D$="CAS:" THEN PRINT "LIGUE O GRA
VADOR E APERTE ENTER":A$=INPUT$(1)
340 OPEN D$+N$ FOR OUTPUT AS 1
350 PRINT #1,M;N:FOR P=1 TO M:PRINT #1,I
(P);J(P):NEXT P:FOR P=1 TO N:PRINT #1,X(
P);Y(P);Z(P):NEXT P:PRINT #1,OX;OY;OZ;R:
CLOSE 1:RETURN
360 CLS:INPUT "NOME DO ARQUIVO ";N$:GOSU
B 470:PRINT "CONFIRMA LEITURA (S,N) ?":A
$=INPUT$(1):IF A$<>"S" AND A$<>"s" THEN
RETURN
370 IF D$="CAS:" THEN PRINT "LIGUE O GRA
VADOR E APERTE ENTER":A$=INPUT$(1)
380 OPEN D$+N$ FOR INPUT AS 1
390 INPUT #1,M,N:FOR P=1 TO M:INPUT #1,I
(P),J(P):NEXT P:FOR P=1 TO N:INPUT #1,X(
P),Y(P),Z(P):NEXT P:INPUT #1,OX,OY,OZ,R:
CLOSE 1:RETURN
400 M=80:N=40:DIM X(N),Y(N),Z(N),A(N),B(
N),C(N),D(N),E(N),F(N),I(M),J(M):RETURN
410 CLS:KEYOFF:SCREEN 0:COLOR 15,4,4
420 PRINT "        ### PERSPECTIVAS CONICA
S ###":PRINT:PRINT:PRINT "OPCOES:":PRINT
:PRINT "1>ENTRAR DADOS":PRINT "2>OBSERVA
DOR":PRINT "3>CALCULAR":PRINT "4>VISUALI
ZAR":PRINT "5>CORRIGIR":PRINT "6>SALVAR
DADOS":PRINT "7>LER DADOS"
430 PRINT:PRINT:INPUT "SUA OPCAO ";A:IF
A<1 OR A>7 THEN 410
440 IF A=1 THEN 40
450 ON A GOSUB 0,110,130,230,250,320,360
460 GOTO 410
470 PRINT "CASSETE OU DISCO (C,D) ?":A$=
INPUT$(1):IF A$="C" OR A$="c" THEN D$="C
AS:":RETURN
```

```
480 IF A$<>"D" AND A$<>"d" THEN 470
490 PRINT "QUAL DRIVE (A,B,C,D) ?":D$=IN
PUT$(1):IF (D$<"A" OR D$>"D") AND (D$<"a
" OR D$>"d") THEN 490 ELSE D$=D$+":":RET
URN
```

ANÁLISE

O programa inicia zerando a variável FL (de "FLAG"), que indica se é possível imprimir a imagem projetada, e sofre um desvio para a linha 410. Desta linha até a 440 é pedida a opção do usuário. Se a opção pedida não existir, ele reinicia o menu.

As linhas 40 a 100 são responsáveis pela opção 1 (entrar dados) onde são dadas as entradas das variáveis do observador e a distância do plano de quadro (opção 2).

O coração do programa situa-se nas linhas 130 a 210, etapa responsável pelos cálculos de geometria analítica. A projeção pode ser própria (observador próximo da origem) ou imprópria (observador muito distante). Uma vez que as coordenadas do observador formam um vetor posição, não se deve situar o observador coincidentemente com a origem, pois isto levaria o programa a usar um vetor nulo, tornando impossível determinar o plano de quadro.

As linhas 220 a 240 são responsáveis pelo desenho da figura projetada na tela de imagem. Se FL valer 0, o desenho é feito; se valor 1, o computador avisará que o desenho sairia da tela e causaria a interrupção do programa.

Se você possui uma impressora gráfica (padrão EPSON) e deseja imprimir nela uma cópia da figura obtida, altere o programa conforme a listagem 2. Uma cópia será automaticamente feita na impressora.

As linhas 250 a 310 servem para se fazer a correção das coordenadas de um ponto escolhido pelo usuário, desde que este ponto exista.

Finalmente, as linhas 320 a 400 permitem armazenar e recuperar os dados de um sólido definido pelo usuário e as coordenadas do observador.

INSTRUÇÕES

Esse programa mostra na tela o calendário de qualquer mês, de qualquer ano compreendido entre 1583 e 2499. Ele mostra também os feriados móveis, tais como a Páscoa e Carnaval, e os feriados fixos, tais como Natal e Dia da Independência.

A utilização do programa Calendário é bastante simples. Após rodá-lo, você deve entrar o ano e o mês desejados.

O único cuidado é quanto à digitação do ano: deve conter os quatro algarismos e estar compreendido entre 1583 e 2499.

Após apresentado na tela o calendário do mês e ano pedidos, a pressão da tecla "1" permite o reinício do programa, para possibilitar a mudança do ano e a pressão da tecla "2" permite a escolha de outro mês do mesmo ano.

Figura 19.1 — Programa CALENDÁRIO

```
100 '  C A L E N D A R I O
110 '  LUIZ TARCISIO DE CARVALHO JR
120 '
130 CLEAR:KEYOFF:COLOR15,4:SCREEN0:WIDTH
35
140 DATAJANEIRO,FEVEREIRO,MARCO,ABRIL,MA
IO,JUNHO,JULHO,AGOSTO,SETEMBRO,OUTUBRO,N
OVEMBRO,DEZEMBRO
150 DIMM$(12):FORU=1TO12:READM$(U):NEXT
160 DATAterca-CARNAVAL,sexta-PAIXAO,quin
ta-CORPUS CHRISTI,1-ANO NOVO,21-TIRADENT
ES,1-DIA DO TRABALHO,7-INDEP.BRASIL,12-N
.SRA.APARECIDA,2-FINADOS,15-PROCL.REPUBL
ICA,25-NATAL
```

```
170 DIMF$(11):FORU=1TO11:READF$(U):NEXT
180 OPEN"GRP:"FOR OUTPUT AS #1
190 LOCATE7,1: PRINT"CALENDARIO PERMANEN
TE"
200 LOCATE2,6:PRINT"(1583 A 2499)
210 LOCATE3,5:INPUT"QUAL O ANO ";AN
220 IF AN < 1583 OR AN > 2499 THEN COLOR
 15,8:CLS:LOCATE 10,10:PRINT"DADO INVALI
DO":FORU=1TO1000:NEXT:RUN
230 BI=0:IFAN/4<>AN\4 THEN 260
240 IFAN/400=AN\400THEN BI=1:GOTO 260
250 IF AN/100<>AN\100 THEN BI=1
260 ND=365+BI:DIMD(ND),N(12),M(13)
270 IF AN>=1583 AND AN<1700 THEN X=22:Y=
2:GOTO380
280 FOR I=17 TO 24:IF AN>=I*100 ANDAN <(
I+1)*100 THEN J=I-16:GOTO 300
290 NEXT I
300 ON J GOTO 310,320,330,330,340,350,36
0,370
310 X=23:Y=3:GOTO380
320 X=23:Y=4:GOTO380
330 X=24:Y=5:GOTO380
340 X=24:Y=6:GOTO380
350 X=25:Y=0:GOTO380
360 X=26:Y=1:GOTO380
370 X=25:Y=1
380 A=ANMOD19:B=ANMOD4:C=ANMOD7:D=(19*A+
X)MOD30:E=(2*B+4*C+6*D+Y)MOD7:P=22+D+E
390 IF P<=31 THEN P=59+BI+P:GOTO420
400 P=D+E-9:IFP<=25THENP=P+BI+90:GOTO420
410 P=P+83+BI
420 CA=P-47:PA=P-2:CC=P+60
430 D1=PMOD7:IFD1=0THEND1=7
440 N(1)=9-D1:IFN(1)=8THENN(1)=1
450 N(2)=N(1)+3:IFN(2)>7THENN(2)=N(2)-7
460 N(3)=N(2)+BI:IFN(3)>7THENN(3)=N(3)-7
470 N(4)=N(3)+3:IFN(4)>7THENN(4)=N(4)-7
480 N(5)=N(4)+2:IFN(5)>7THENN(5)=N(5)-7
490 N(6)=N(5)+3:IFN(6)>7THENN(6)=N(6)-7
500 N(7)=N(6)+2:IFN(7)>7THENN(7)=N(7)-7
```

```
510 N(8)=N(7)+3:IFN(8)>7THENN(8)=N(8)-7
520 N(9)=N(8)+3:IFN(9)>7THENN(9)=N(9)-7
530 N(10)=N(9)+2:IFN(10)>7THENN(10)=N(10
)-7
540 N(11)=N(10)+3:IFN(11)>7THENN(11)=N(1
1)-7
550 N(12)=N(11)+2:IFN(12)>7THENN(12)=N(1
2)-7
560 FOR I=D1 TO ND STEP 7:D(I)=12:NEXT
570 D(CA)=1:D(PA)=2:D(CC)=3
580 D(1)=4:D(111+BI)=5:D(121+BI)=6:D(250
+BI)=7:D(285+BI)=8:D(306+BI)=9:D(319+BI)
=10:D(359+BI)=11
590 LOCATE4,11:PRINT"(1 A 12)"
600 LOCATE3,10:INPUT"QUAL O MES ";ME
610 IFME<1ORME>12ORME<>INT(ME)THENCOLOR1
5,8:CLS:LOCATE10,10:PRINT"DADO INVALIDO"
:FORU=1TO1000:NEXT:COLOR15,4:CLS:LOCATE7
,1:PRINT"CALENDARIO PERMANENTE":LOCATE3,
5:PRINT"ANO =";AN:GOTO590
620 COLOR15,1,1:SCREEN2
630 COLOR3:PSET(INT(256-8*(LEN(M$(ME))+8
))/2,8),1:PRINT#1,M$(ME);" DE";AN
640 COLOR7:PSET(0,24),1:PRINT#1,"   DOM
SEG TER QUA QUI SEX SAB  "
650 DATA0,31,59,90,120,151,181,212,243,2
73,304,334,365
660 RESTORE650
670 FORU=1TO13:READM(U):NEXT
680 Q=0:IF ME=2THEN Q=BI
690 S=0:IFME>2THENS=BI
700 COLOR14:PSET(0,40),1:FORU=1TON(ME)*4
:PRINT#1," ";:NEXTU
710 FORI=M(ME)+1TOM(ME)+8-N(ME)
720 IFD(I+S)<>0THENCOLOR8:PRINT#1,I-M(ME
);" ";:COLOR14:GOTO740
730 PRINT#1,I-M(ME);" ";
740 NEXTI
750 W=40:FORJ=ITOM(ME+1)+QSTEP7
760 W=W+16:PSET(0,W),1:PRINT#1,"    ";:IF
J-M(ME)<10THENPRINT#1," ";
```

```
770 FORK=JTOJ+6:IFK>M(ME+1)+QTHENGOTO800
780 IFD(K+S)<>0THENCOLOR8:PRINT#1,K-M(ME
);:COLOR14:GOTO800
790 PRINT#1,K-M(ME);
800 IFK-M(ME)<9THENPRINT#1," ";
810 NEXTK
820 NEXTJ
830 W=W+16:PSET(0,W),1:COLOR3:PRINT#1,ST
RING$(32,223);
840 FORI=M(ME)+1TOM(ME+1)+Q
850 IFD(I)>0ANDD(I)<12THENW=W+10:PSET(30
,W),1:COLOR13:PRINT#1,F$(D(I))
860 NEXTI
870 W=W+8:PSET(0,W),1:COLOR3:PRINT#1,STR
ING$(32,220);
880 PSET(22,180),1:COLOR3:PRINT#1,"[1]OU
TRO ANO / [2]OUTRO MES"
890 A$=INKEY$
900 IFA$="1"THENRUN
910 IFA$="2"THENGOTO930
920 GOTO890
930 COLOR15,4:SCREEN0:LOCATE7,1:PRINT"CA
LENDARIO PERMANENTE":LOCATE3,5:PRINT"ANO
 =";AN:GOTO590
```

ANÁLISE

O programa está inteiramente baseado no algoritmo de Gauss que se refere ao cálculo da data da Páscoa.

Se a data da Páscoa for determinada num certo ano, saberemos calcular as datas de todos os outros feriados móveis, bem como em que dia da semana caem todos os dias desse ano, uma vez que a Páscoa sempre acontece num domingo.

As linhas 130 a 170 definem as matrizes que conterão os nomes dos meses (M$) e os nomes dos feriados (F$).

As linhas 190 a 210 permitem a entrada do ano desejado. Na linha 220 verifica-se se a entrada é válida, caso contrário, o programa "roda" novamente.

As linhas 230 a 250 verificam se o ano pedido é bissexto. Se for, é atribuído variável B1 o valor 1.

A linha 260 dimensiona uma matriz (ND) cujos elementos deverão corresponder aos dias do ano. Se a qualquer um desses elementos for atribuído um valor diferente de zero, significa que o dia a ele correspondente deve ser um feriado ou um domingo.

Além disso, são também dimensionadas duas outras matrizes: uma que conterá a informação sobre o dia da semana em que cai o primeiro dia de cada mês do ano (N(12)), e outra que guardará o número de dias decorridos desde o início do ano até o primeiro dia de cada mês (M(13)).

As linhas 270 a 410 contêm o algoritmo de Gauss e determinam o dia do ano em que cai a Páscoa (P). A seguir, na linha 420, são calculados os dias que correspondem aos demais feriados móveis: Terça-feira de Carnaval (CA), Paixão (PA) e Corpus Christi (CC).

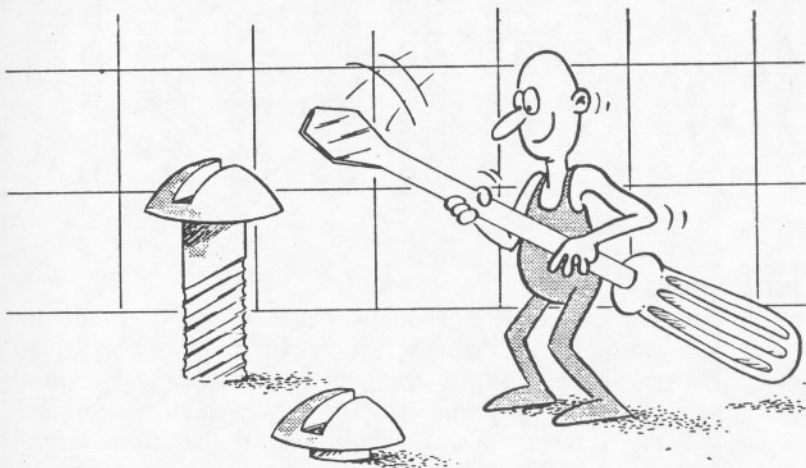
As linhas 430 a 550 calculam o primeiro dia de cada mês e as de número 560 a 580 calculam os feriados e domingos.

Finalmente, as linhas 90 a 610 permitem a entrada do mês e as demais "imprimem" na SCREEN 2 o calendário do mês pedido (figura 19.2).

Figura 19.2 — Folhinha do mês

```
        DEZEMBRO DE 1987
   DOM SEG TER QUA QUI SEX SAB
                 1   2   3   4   5
     6   7   8   9  10  11  12
    13  14  15  16  17  18  19
    20  21  22  23  24  25  26
    27  28  29  30  31
  ________________________________
    25-NATAL
  ________________________________

   [1]OUTRO ANO / [2]OUTRO MES
```

UTILITÁRIOS

Os usuários mais experientes, que dominam com maestria a arte da programação, normalmente se utilizam de rotinas que os ajudam na elaboração e concretização de outros programas.

Esses "meta-programas" são chamados de UTILITÁRIOS e, apesar de não ser estritamente necessário, normalmente, são codificados em linguagem de máquina para serem mais rápidos, mais curtos e mais eficazes.

Nas próximas páginas estaremos "fechando" este segundo volume do COLEÇÃO DE PROGRAMAS com chave de ouro: 6 programas utilitários de alto-nível, desenvolvidos para nosso próprio uso e explicados em seus aspectos mais importantes.

Três deles se destinam à otimização do uso da impressora: um permite obter cópias de telas gráficas e outros dois geram padrões pré-definidos e ampliados. Lembre-se de alterar o endereço &HF417 de acordo com a impressora que será usada. Paradoxalmente (pasmem!) os MSX nacionais em suas versões mais recentes NÃO conectam de forma imediata com impressoras padrão MSX! Veja no manual do seu micro, o valor a ser inserido no endereço &HF417 (0 ou 1) se houver algum problema com os programas que usam impressora.

Entre os outros programas temos um "vasculhador de bytes", um "abridor" de programas em fita e um setorizador de tela para a SCREEN 1.

DUMP DE MEMÓRIA

```
7F 05 C3 D7 07 C3 CD 07 048
C3 EC 07 C3 DF 07 C3 15 050
08 C3 0F 07 C3 44 07 C3 058
4F 08 C3 F7 07 00 C3 98 060
13 C3 A8 06 C3 0E 05 C3 068
38 05 C3 D2 05 C3 1F 06 070
C3 94 05 C3 B4 05 C3 02 078
```

INSTRUÇÕES

Muitas vezes precisamos analisar uma região da memória para saber porquê certas coisas acontecem de uma certa maneira. Os programas que mostram os dados gravados na memória são chamados de programas DUMP, e é justamente isso o que faz este nosso programa: mostra uma região da memória do micro no formato hexadecimal e seus respectivos caracteres.

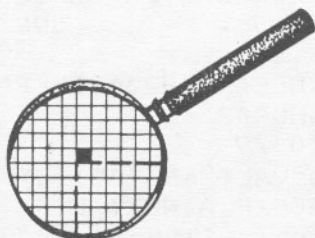
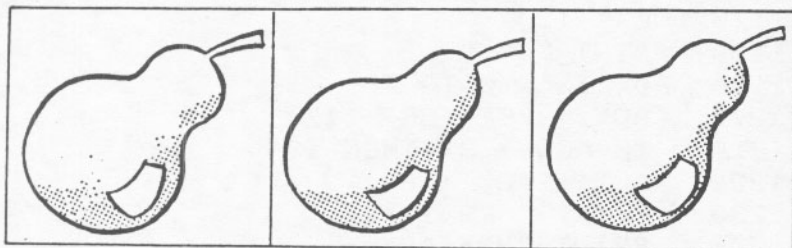
O programa pergunta qual o endereço da memória que será analisado, devendo este ser dado em decimal ou então em hexadecimal, desde que precedido por &H. A partir daí são mostrados 128 bytes de memória e novamente é pedido um outro endereço. Neste ponto pode-se digitar apenas RETURN que serão mostrados os 128 bytes seguintes.

Figura 20.1 — Programa DUMP de MEMÓRIA

```
1000 REM DUMP DE MEMORIA
1010 REM Rubens Jr.   JUN/86
1020 REM
1030 SCREEN 0 : WIDTH 40 : KEY OFF
1040 COLOR 15,1,1 : CLS
1050 INPUT "Endereco ";ED
1060 LOCATE 0,0
1070 PRINT "Ender ---------- hex";
1080 PRINT " ----------   ASCII "
1090 PRINT
1100 FOR ED = ED TO ED + 120 STEP 8
1110  EX$ = RIGHT$("0000"+HEX$(ED),4)
1120  PRINT EX$;": ";
1130  FOR I% = 0 TO 7
1140   BY% = PEEK(ED + I%)
```

```
1150    BX$ = RIGHT$("00"+HEX$(BY%),2)
1160    PRINT BX$;" ";
1170   NEXT I%
1180   PRINT " ";
1190   FOR I% = 0 TO 7
1200    AC% = PEEK(ED + I%)
1210    IF AC% > 31 THEN 1240
1220    AC% = AC% + 64
1230    PRINT CHR$(1);
1240    PRINT CHR$(AC%);
1250   NEXT I%
1260   PRINT
1270  NEXT ED
1280  PRINT
1290  GOTO 1050
```

ANÁLISE

O programa tem apenas duas partes principais: a que mostra os dados em hexadecimal! (linhas 1150 a 1190) e a que mostra os caracteres (linhas 1210 a 1270). As demais linhas apenas completam o programa, não tendo portanto nada de especial ou de difícil para ser entendido. Mesmo assim, vale a pena observar as linhas 1130 e 1170 que ajustam um valor em hexa para terem sempre quatro ou dois dígitos respectivamente, independentemente do seu valor. Outro macete também está nas linhas 1240 a 1260 que mostram os caracteres de código menores que 32. Como já comentamos no Coleção de Programas vol. I, os caracteres de código menores que 32 só podem ser impressos após um PRINT CHRS(1) e com seus códigos acrescidos de 64 unidades.

GIRO PARCIAL DE TELA

INSTRUÇÕES

Este programa tem como objetivo rolar algumas linhas da SCREEN 1 para baixo, facilitando a digitação de dados em tabelas.

Para se conseguir isso, foi utilizada uma sub-rotina em Linguagem de Máquina para se obter uma alta velocidade de execução (se comparada com outras linguagens de alto nível).

Uma vez carregada esta sub-rotina, é necessário informar dois parâmetros antes dela ser executada. Estes dois parâmetros são respectivamente a linha superior e a linha inferior que limitam a região do vídeo que será rolada. A linha inferior será perdida e a linha superior será preenchida com brancos.

Tome cuidado ao informar estas duas linhas, pois a sub-rotina não verifica se são válidas ou não, e se forem dados valores incorretos o micro pode ficar sem controle!!!

A linha superior é informada pelo endereço &HDA00 e a linha inferior pelo endereço&HDA01. O início desta sub-rotina é em &HDA02.

Figura 21.1 — Programa GIRO PARCIAL DA SCREEN 1

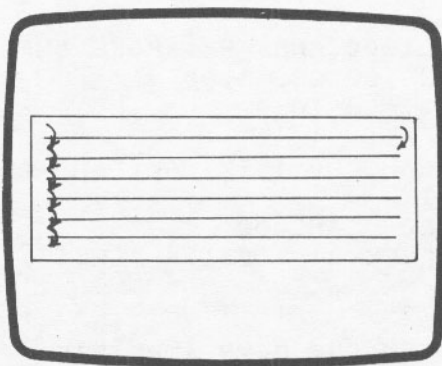
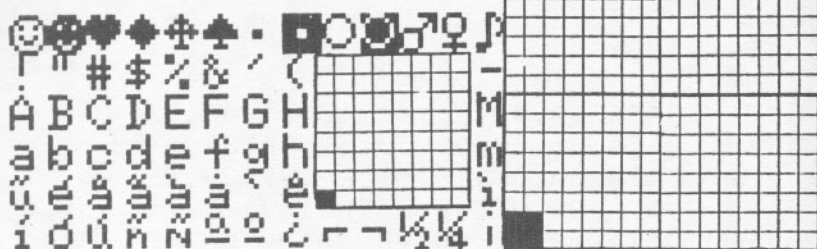
```
1000 REM GIRO PARCIAL DA SCREEN 1
1010 REM Rubens Jr.           JUN/86
1020 REM
1030 SCREEN 1:CLEAR &H200,&HCFFF
1040 DEFUSR0=&HDA02
1050 FOR F=0 TO 79
1060 READ A$:A=VAL("&H"+A$)
1070 POKE &HDA00+F,A:NEXT F
1080 INPUT"Entre a linha inicial";LI
1090 INPUT"Entre a linha final";LF
```

```
1100 POKE &HDA00,LI:POKE &HDA01,LF
1110 LOCATE 0,LI
1120 INPUT"Entre uma mensagem";A$
1130 A=USR(0)
1140 GOTO 1110
1150  DATA 00,00,21,00,18,11,4F,DA
1160  DATA 01,00,03,CD,59,00,3A,01
1170  DATA DA,6F,26,00,29,29,29,29
1180  DATA 29,01,4F,DA,0B,09,EB,21
1190  DATA 20,00,19,EB,3A,00,DA,47
1200  DATA 3A,01,DA,90,E5,6F,26,00
1210  DATA 29,29,29,29,29,44,4D,E1
1220  DATA ED,B8,23,06,20,36,20,23
1230  DATA 10,FB,21,4F,DA,11,00,18
1240  DATA 01,00,03,CD,5C,00,C9,00
```

ANÁLISE

A sub-rotina em si está nas linhas DATA de 1150 a 1240.

As linhas de 1030 a 1070 apenas a carrega na memória do micro e devem ser as primeiras linhas do programa onde esta sub-rotina será usada devido à presença do comando CLEAR na linha 1030 que é fundamental para o seu perfeito funcionamento.

As linhas de 1080 a 1140 são apenas um exemplo de como usar esta sub-rotina e o efeito que ela produz, devendo ser substituídas pelo verdadeiro programa que a utilizará.

CARACTERES 16 x 16

INSTRUÇÕES

Este programa imprime os caracteres com tamanho de 16 por 16 pontos utilizando o modo gráfico da impressora, funcionando portanto apenas se você possue uma impressora gráfica e esta entrar no modo gráfico segundo o padrão EPSON.

Pode-se imprimir qualquer caractere, inclusive os gráficos. Cada caracter tem seu formato pesquisado na ROM e expandido dos 8 X 8 para 16 X 16.

Pelo fato dos micros MSX interpretarem certos caracteres quando eles são enviados para a impressora, foi utilizada uma pequena sub-rotina em Linguagem de Máquina para contornar tal inconveniente.

Figura 22.1 — Programa CARACTERES 16 x 16

```
1000 REM CARACTERES  16 X 16
1010 REM
1020 REM                Rubens Jr.   JUN/86
1030 REM
1040 CLEAR 1000,&HBFFF : POKE &HF417,1
1050 DATA 3E,00,C3,A5,00
1060 FOR I% = 0 TO 4
1070  READ A$
1080  POKE &HC000 + I%,VAL("&H"+A$)
1090 NEXT I%
1100 DEFUSR0 = &HC000
1110 ET = PEEK(4) + 256 * PEEK(5)
1120 REM
1130 TX$ = ""
1140 INPUT "O que devo imprimir ";TX$
1150 IF TX$ = "" THEN CLS : END
1160 TM% = 0
```

```
1170 FOR I% = 1 TO LEN(TX$)
1180  X% = ASC(MID$(TX$,I%,1))
1190  IF X% <> 1 THEN TM% = TM% + 16
1200 NEXT I%
1210 HI% = TM% \   256
1220 LO% = TM% MOD 256
1230 LPRINT CHR$(27);"A";CHR$(8);
1240 REM
1250 FOR I% = 0 TO 1
1260  LPRINT CHR$(27);"K";
1270  POKE &HC001,LO%
1280  X = USR0(0)
1290  POKE &HC001,HI%
1300  X = USR0(0)
1310  LT% = 1
1320  AC% = ASC(MID$(TX$,LT%,1))
1330  IF AC% <> 1 THEN 1350
1340  AC%=ASC(MID$(TX$,LT%+1,1))-64
1350  FOR CO% = 0 TO 7
1360   AG% = 0
1370   FOR LI% = 0 TO 3
1380    ED = 8*AC%+ET+4*I%+LI%
1390    BY% = PEEK(ED)
1400    BY% = BY% AND (128/(2^CO%))
1410    IF BY% THEN BY% = 1
1420    BT% = 128/(4^LI%)
1430    BT% = BT%+(BT%/2)
1440    AG% = AG%+(BY%*BT%)
1450   NEXT LI%
1460   POKE &HC001,AG%
1470   X% = USR0(0)
1480   X% = USR0(0)
1490  NEXT CO%
1500  AC% = ASC(MID$(TX$,LT%,1))
1510  IF AC% = 1 THEN LT% = LT%+1
1520  LT% = LT%+1
1530  IF LT% <= LEN(TX$) THEN GOTO 1320
1540  LPRINT
1550 NEXT I%
1560 GOTO 1140
```

ANÁLISE

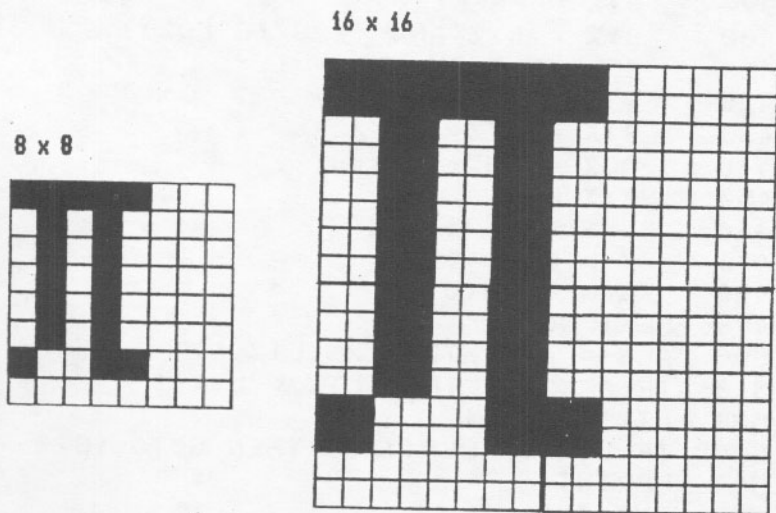
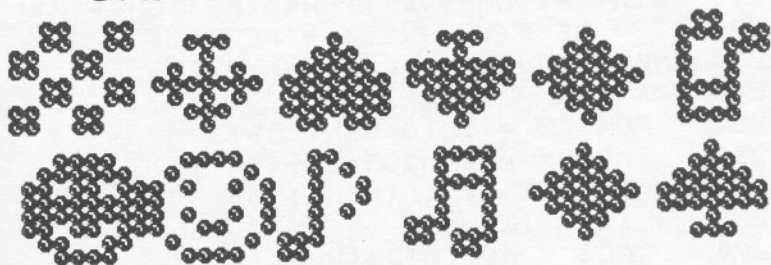
No início do programa é carregada a sub-rotina em Linguagem de Máquina como normalmente é feito através dos comandos READ e DATA.

O texto a ser impresso é pedido nas linhas 1130 a 1150, onde foi usado um pequeno truque para poder se encerrar o programa apenas teclando RETURN. Com o texto já digitado, é calculado quantos caracteres serão enviados impressora nas linhas 1160 a 1220.

Na linha 1230 a impressora é ajustada para pular de 8 em 8 pontos ao término de cada linha.

Nas demais linhas são enviados os caracteres um a um, sendo enviado primeiramente a metade superior e depois a metade inferior de cada caractere, já que não é possível imprimir 16 pontos de uma só vez.

Por ser necessário rodar a matriz de 8 X 8 pontos da ROM para 16 X 16 o programa faz certos cálculos um tanto quanto complicados, fazendo sua execução ficar um tanto lenta.

Figura 22.2 — O caractere π em sua forma 8 x 8 e 16 x 16

CARACTERES GIGANTES

INSTRUÇÕES

Este programa imprime os caracteres com tamanho de 8 por 8 pontos, onde cada ponto tem seu formato definido por você. Se você não possuir uma impressora gráfica padrão EPSON, infelizmente não poderá ver este programa ser executado com toda a sua potencialidade, mas poderá ter uma idéia do que ele faz utilizando o vídeo.

Figura 23.1 — Programa CARACTERES GIGANTES

```
1000 REM Caracteres Gigantes
1010 REM Rubens Jr.   JUN/86
1020 REM
1030 REM definicao do caractere
1040 DATA 00111100
1050 DATA 01111010
1060 DATA 11011111
1070 DATA 11011111
1080 DATA 11100111
1090 DATA 11111111
1100 DATA 01111110
1110 DATA 00111100
1120 A$="" : POKE &HF417,1
1130 FOR F=1 TO 8
1140   READ B$:A$=A$+CHR$(VAL("&b"+B$))
1150 NEXT F
1160 REM inicializacao da impressora
1170 BR$=CHR$(27)+"K"+CHR$(8)+CHR$(0)+ST
RING$(8,0)
1180 PR$=CHR$(27)+"K"+CHR$(8)+CHR$(0)+A$
1190 REM impressao da mensagem
```

```
1200 OPEN "lpt:" FOR OUTPUT AS #1
1210 PRINT #1,CHR$(27);CHR$(65);CHR$(7)
1220 ET = PEEK(4) + 256 * PEEK(5)
1230 INPUT "Qual a mensagem ";CP$
1240 FOR RW% = 0 TO 7
1250   FOR I% = 1 TO LEN(CP$)
1260     AC% = ASC(MID$(CP$,I%,1))
1270     IF AC% <> 1 THEN 1300
1280     I% = I% + 1
1290     AC% = ASC(MID$(CP$,I%,1)) - 64
1300     BT$ = BIN$(PEEK(AC%*8+ET+RW%))
1310     BT$ = RIGHT$("0000000"+BT$,8)
1320     FOR J% = 1 TO 8
1330      X$ = MID$(BT$,J%,1)
1340      IF X$ = "0" THEN PRINT #1,BR$;
1350      IF X$ = "1" THEN PRINT #1,PR$;
1360     NEXT J%
1370    NEXT I%
1380    PRINT #1," "
1390 NEXT RW%
1400 GOTO 1230
```

ANÁLISE

No início do programa são definidos duas variáveis principais, BR$ e PR$ que são os formatos dos pontos apagados e acesos.

O formato de cada caractere é pesquisado na própria ROM do micro.

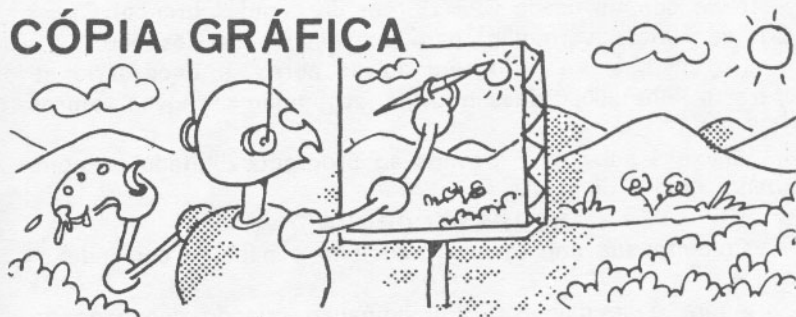
Paraque o programa ficasse bem flexível, foi utilizado um arquivo de saída em vez de se utilizar os comandos próprios para a impressora. Assim, se você não possuir uma, elimine as linhas de 1030 a 1180 e a linha 1210 também. Insira em seguida as seguintes linhas:

```
1030 BR$ = " "
1040 PR$ = "*"
```

E modifique também a linha 1200 para:

```
1200 OPEN"CRT:" FOR OUTPUT AS #1
```

Estas modificações visam o uso do vídeo em vez da impressora.

CÓPIA GRÁFICA

INSTRUÇÕES

O MSX é um microcomputador com excelentes recursos gráficos. Consequentemente existe uma grande quantidade de usuários que o utilizam para fazer desenhos, trabalhando principalmente na SCREEN 2.

Para estes usuários, desenvolvemos uma sub-rotina que copia a SCREEN 2 na impressora, bit a bit. Para tanto, é necessário uma impressora que tenha recursos gráficos. O programa a seguir funciona, por exemplo, numa Mônica da Elebra, que entra no modo gráfico segundo o padrão EPSON.

Digite a sub-rotina que vai da linha 50000 à 50200. Se você tiver um disk-drive trabalhando com o DISK-BASIC, grave-a com o comando:

```
SAVE"COPGRA.ASC",A
```

Para gravá-la em cassete, comande:

```
SAVE"CAS:COPGRA"
```

Desta forma a rotina fica armazenada em ASC, permitia sua anexação ao programa que gera o desenho através do MERGE.

Para experimentar seu funcionamento, digite o programinha a seguir:

Figura 24.1 — Teste da Cópia Gráfica

```
10 CP=15:CT=8:COLOR CT,CP:SCREEN 2
20 CIRCLE(128,96),80,CT:PAINT(128,96),CT
30 FOR I=0 TO 255:PRESET(I,96):NEXT I
40 GOSUB 50000:END
```

Note que foi usado CP=15 (cor do papel = branco) CT=8 (Cor de Tinta = vermelho) para a execução do desenho.

No caso de se utilizarem outras cores, é necessário se alterar a linha 50000 da sub-rotina, atribuindo os novos valores a CP e CT.

Anexe a sub-rotina gravada ao programa digitado no computador com o comando:

```
MERGE "COPGRA"
```

Conecte sua impressora, posicione o papel e comande:

```
RUN
```

Agora é só dar asas à imaginação criando desenhos na SCREEN 2, registrando-os no papel.

Figura 24.2 — Programa CÓPIA GRÁFICA

```
50000 POKE &HF417,1:CP=15:CT=8
50010 DATA 3E,00,CD,A5,00,C9
50020 FOR I = 0 TO 5:READ A$:POKE 51000!
+I,VAL("&H"+A$):NEXT I:DEFUSR0=51000!
50030 LPRINT CHR$(27);CHR$(65);CHR$(8)
50040 FOR C=0 TO 31
50050 LPRINT CHR$(27);CHR$(75);CHR$(192)
;CHR$(0);
50060 FOR L=23 TO 0 STEP -1
50070 FOR X=7 TO 0 STEP -1
50080 U=VPEEK((C*8+256*L)+X)
50090 V=VPEEK((C*8+256*L)+X+8192)
50100 IF V MOD 16 =CT  THEN U=255
50110 IF V\16=CP THEN U=0
50120 POKE 51001!,U
50130 Z=USR0(0)
50140 NEXT X
50150 NEXT L
50160 LPRINT CHR$(10);
50170 NEXT C
50180 LPRINT CHR$(27);CHR$(65);CHR$(13)
50190 LPRINT CHR$(27);CHR$(81);CHR$(39)
50200 RETURN
```

ANÁLISE

A linha 50000 define quais as cores utilizadas para a tinta (CT) e o papel (CP). A instrucão:

```
POKE &HF417,1
```

só é necessária no HOTBIT ou no EXPERT versão 1.1, pois tira o computador no modo ABNT, permitindo que ele comande a MÔNICA segundo o padrão EPSON.

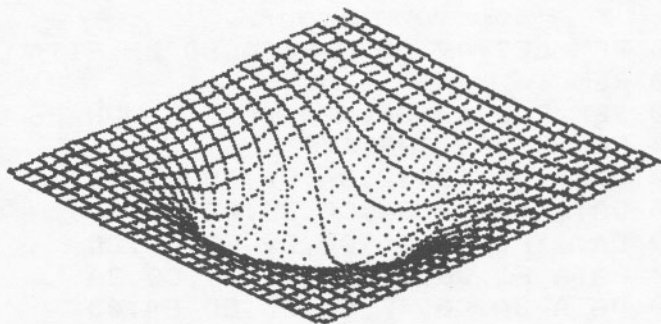
As linhas 50010 e 50020 inserem um curto programa em linguagem de máquina a partir do endereço 51000 da RAM. Este programinha será chamado pela linha 50130 quando for necessário se enviar um byte para a impressora.

A linha 50030 regula o avanço de papel da impressora de 8 em 8 pontos de maneira que cada "tira" gráfica seja impressa imediatamente abaixo da anterior.

As linhas de 50040 a 50170 fazem a leitura da VRAM e transmitem as informações obtidas à impressora.

A linha 50180 recoloca o avanço do papel da impressora em seu valor normal (13) e a linha 50190 regula a largura da impressão em 40 colunas. Isto é conveniente para se tirarem listagens com LLIST pois seu aspecto no papel ficará muito parecido com o da tela na SCREEN 0, permitindo uma melhor comparação. Aliás, todas as listagens deste volume foram tirads segundo este padrão.

Como se trata de uma sub-rotina, a última linha (50200) não podia deixar de conter um RETURN.

Para quem quiser os detalhes de como o "miolo" desta rotina faz a leitura da VRAM, aconselhamos a leitura do "APROFUNDANDO-SE NA MSX" desta mesma editora.

MONSIEUR LEFITTA

INSTRUÇÕES

Muitas vezes queremos saber em quais endereços de memória um programa em Linguagem de Máquina é carregado, e com os comandos que o BASIC oferece não é possível descobrí-los.

O programa apresentado a seguir lê uma parte de um programa gravado em fita e mostra seu nome, de que tipo ele é (Linguagem de Máquina, BASIC ou ARQUIVO) e, se ele estiver em Linguagem de Máquina, mostra também seus endereços de início, fim e de execução.

Para conseguir isso, foi usada uma pequena sub-rotina em Linguagem de Máquina que utiliza algumas rotinas do próprio BIOS do MSX.

Todos os dados gravados em fita cassete, sejam arquivos ou programas de qualquer tipo, tem um "cabeçalho" inicial, onde reside o nome e o tipo do programa. Logo em seguida, vem um outro bloco de dados contendo alguns parâmetros e só depois está o programa propriamente dito.

Figura 25.1 — Programa MONSIEUR LEFITTA

```
1000 REM LEITOR DE CABECALHO DE FITA
1010 REM
1020 REM Rubens Jr.                JUN/86
1030 CLEAR 200,&HBFFF
1040 DATA 3E,10,32,3A,C0,18,05,3E
1050 DATA 06,32,3A,C0,3E,02,32,63
1060 DATA F6,21,00,00,22,F8,F7,CD
1070 DATA E1,00,38,1A,21,3B,C0,3A
1080 DATA 3A,C0,47,E5,C5,CD,E4,00
1090 DATA C1,E1,38,0A,77,23,10,F3
1100 DATA 21,FF,FF,22,F8,F7,CD,E7
1110 DATA 00,C9,00,00,00,00,00,00
```

```
1120 FOR I = &HC000 TO &HC03A
1130  READ X$
1140  POKE I,VAL("&H"+X$)
1150 NEXT I
1160 DEFUSR0=&HC000 : DEFUSR1=&HC007
1170 DEF FNP(X)=PEEK(X)+256*PEEK(X+1)
1180 AP$ = CHR$(34)
1190 SCREEN 0 : WIDTH 40 : COLOR 15,1
1200 KEY OFF  : CLS
1210 LOCATE 11,0,0
1220 PRINT "LEITOR DE CABECALHO"
1230 LOCATE 11,1
1240 PRINT "==================="
1250 LOCATE 0,5
1260 PRINT "Prepare o gravador e tecle
<RETURN> ";
1270 X$ = INPUT$(1)
1280 LOCATE 0,5
1290 PRINT "Aguarde, Lendo o nome do pro
grama.   ";
1300 IF USR(0) = 0 THEN 1770
1310 LOCATE 0,5
1320 PRINT SPC(37);
1330 LOCATE 0,4
1340 PRINT "Programa ... : ";
1350 NM$ = ""
1360 FOR I = &HC045 TO &HC04A
1370  NM$ = NM$ + CHR$(PEEK(I))
1380 NEXT I
1390 PRINT NM$
1400 LOCATE 0,6
1410 PRINT "Formato .... : ";
1420 TP = PEEK(&HC03B)
1430 IF TP = &HD0 THEN 1470
1440 IF TP = &HD3 THEN 1730
1450 IF TP = &HEA THEN 1800
1460 GOTO 1840
1470 REM FORMATO BLOAD
1480 PRINT "BINARIO"
1490 LOCATE 0,9
1500 PRINT "Aguarde, lendo dados iniciai
s ";
```

```
1510 IF USR1(0) = 0 THEN 1770
1520 LOCATE 0,9
1530 PRINT SPC(29);
1540 LOCATE 0,8
1550 PRINT "Inicio  .... : &H";
1560 PRINT HEX$(FNP(&HC03B))
1570 PRINT
1580 PRINT "Fim ......... : &H";
1590 PRINT HEX$(FNP(&HC03D))
1600 PRINT
1610 PRINT "Execucao ... : &H";
1620 PRINT HEX$(FNP(&HC03F))
1630 PRINT : PRINT
1640 PRINT "Para copiar o programa : "
1650 PRINT
1660 PRINT "BLOAD ";AP$;"CAS:";AP$
1670 PRINT : PRINT
1680 PRINT "BSAVE ";AP$;"CAS:";NM$;AP$;"
,";
1690 PRINT "&H";HEX$(FNP(&HC03B));",";
1700 PRINT "&H";HEX$(FNP(&HC03D));",";
1710 PRINT "&H";HEX$(FNP(&HC03F))
1720 END
1730 REM FORMATO CLOAD
1740 PRINT "BASIC"
1750 PRINT : PRINT
1760 END
1770 LOCATE 0,20
1780 PRINT "* Erro durante a leitura *"
1790 END
1800 REM FORMATO ARQUIVO
1810 PRINT "BASIC"
1820 PRINT : PRINT
1830 END
1840 REM FORMATO NAO PADRAO
1850 PRINT "NAO PADRAO"
1860 PRINT : PRINT
1870 END
```

ANÁLISE

O programa em Linguagem de Máquina está armazenado nas linhas DATA (1040 a 1110) como já é de costume. Após a leitura do primeiro bloco de dados, é mostrado o nome do programa (linhas 1190 a 1390). Dependendo do que foi lido da fita, o programa desvia para rotinas que mostram o tipo do programa lido.

Quando o programa lido estiver em Linguagem de Máquina, serão mostradas duas linhas, uma com o comando BLOAD e a outra com o comando BSAVE. Isso serve para facilitar o trabalho de cópia dos programas de um fita para outra, bastando apenas posicionar o cursor nestas linhas e teclar (RETURN).

Vamos ver uma situação realística. Suponha que um de seus amigos tenha um MSX e possua uma fita com o programa THEZEUS (uma espécie de labirinto, com armadilhas, princesas, robôs, etc...). Se você quiser tirar uma cópia dele numa outra fita, proceda como segue:

1) Digite o programa Monsieur Lefitta.
2) Prepare o gravador com a fita do seu amigo.
3) Tecle F5 para executar o Monsieur Lefitta. O programa pedirá que você coloque o gravador para funcionar (com o PLAY) e que digite RETURN. A seguir, serão apresentadas na tela algumas informações indicando que o programa está gravado com o nome TH e em Linguagem de Máquina (BINÁRIO). Seu endereço inicial é &H8000, seu endereço final é &HBFFF e seu ponto de execução é &H8023.
4) Mova o cursor até a linha com o BLOAD "CAS": prepare o gravador novamente (retornando a fita ao seu início e pressionando PLAY) e digite RETURN.
5) Após o "Ok" surgir na tela, o cursor estará sobre a linha com o BSAVE. Coloque, então, a sua fita no gravador e posicione-a de modo que a parte inicial sem material magnético não seja usada.
6) Pressione PLAY e REC (ou a tecla SAVE, no DATA CORDER) e digite RETURN. Com isso sua fita estará recebendo o programa THEZEUS!
7) Se você tentar copiar um programa muito longo, poderá não caber na memória. Nesse caso, antes de comandar BLOAD, apague o Lefitta e comande NEW.

Editora Aleph

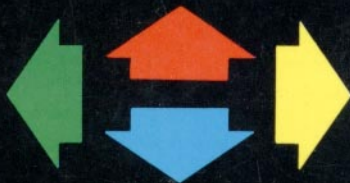
Se você quiser receber gratuitamente informações sobre nossas publicações, preencha o cupom abaixo (ou uma cópia dele) e envie-o para nossa caixa postal.

NOME: ....
RUA/AV.: ....
CEP: .... CIDADE: .... UF: ......
MICRO(s) QUE POSSUI: ....
ÁREA DE INTERESSE: ....

ALEPH P.A.P. Ltda.
Caixa Postal: 20707
CEP: 01498 SP

----

impresso na
Gráfica Palas Athena
Associação "Palas Athena" do Brasil
Rua José Bento, 384
Cambuci - São Paulo
Fone: 279-6288 - CEP: 01523

COLEÇÃO MSX

Dando prosseguimento à COLEÇÃO MSX, a EDITORA ALEPH lança agora o 2º volume da COLEÇÃO DE PROGRAMAS PARA MSX. Mantendo a mesma clareza e profundidade do volume I, este livro incorpora programas aplicativos para a área comercial e financeira, utilitários para programadores avançados, programas didáticos para professores e alunos, jogos de inteligência e de ação, além de muitas dicas e truques para otimização de programas em BASIC e pequenas rotinas úteis em Linguagem de Máquina. Indispensável para aqueles que querem aprender a programar e a usar toda a capacidade das máquinas MSX.